ANÁLISIS DE DESEMPEÑO DE UNA SDEON AL IMPLEMENTAR UN ALGORITMO DE ENRUTAMIENTO DINÁMICO BASADO EN TOPOLOGÍAS VIRTUALES

Cristian Camilo Arguello Guevara
Jhon Alexander Muñoz Lache

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones Programa de Ingeniería Electrónica y Telecomunicaciones
Grupo de Investigación de Nuevas Tecnologías en Telecomunicaciones
GNTT
Popayán, Cauca
2022

ANÁLISIS DE DESEMPEÑO DE UNA SDEON AL IMPLEMENTAR UN ALGORITMO DE ENRUTAMIENTO DINÁMICO BASADO EN TOPOLOGÍAS VIRTUALES

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Cristian Camilo Arguello Guevara
Jhon Alexander Muñoz Lache

Director: I.E. José Giovanny López Perafán PhD.

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones Programa de Ingeniería Electrónica y Telecomunicaciones
Grupo de Investigación de Nuevas Tecnologías en Telecomunicaciones GNTT
Popayán, Cauca
2022

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ECUACIONES

LISTA DE ACRÓNIMOS

AP	*Access Point*, Punto de Acceso.
CHRON	*Cognitive Heterogeneous Reconfigurable Optical Network*, Red Optica Cognitiva Heterogénea Reconfigurable.
CPU	*Central Processing Unit*, Unidad Central de Procesamiento.
EON	*Elastic Optical Networking*, Redes Ópticas Elásticas.
GMPLS	*General Multiprotocol Label Switching*, Conmutación de Etiquetas Multiprotocolo Generalizado.
GUI	*Graphical User Interface*, Interfaz Gráfica de Usuario.
AI	*Artificial Intelligence*, Inteligencia Artificial.
INT	*In-band Network Telemetry*, Telemetría de Red en Banda.
IP	*Internet Protocol*, Protocolo de Internet.
LSP	*Label Switched Paths*, Ruta de Etiquetas Conmutadas.
MF-OTP	*Multi-flow Optical Transponder*, Transpondedor Óptico de Flujo Múltiple.
NFV	*Network Function Virtualization*, Virtualización de Funciones de Red.
NSFNET	*National Science Foundation Network*, Red de la Fundación Nacional para la Ciencia.
ONOS	*Open Network Operating System*, Sistema Operativo de Red Abierta.
PCE	*Path Computation Element*, Elemento de Cálculo de Ruta.
QoS	*Quality of Service*, Calidad de Servicio.
RAM	*Random Access Memory*, Memoria de Acceso Aleatorio.
RMSA	*Routing, Modulation and Spectrum Assignment*, Enrutamiento, modulación y Asignación de Espectro.
RSA	*Routing and Spectrum Allocation*, Enrutamiento y Asignación de Espectro.
SDON	*Software Defined Optical Networking*, Redes Ópticas Definidas por Software.
SDN	*Software Defined Networking*, Redes Definidas por Software.
PCE	*Path Computation Element*, Elemento de Cálculo de Ruta.
UDP	*User Datagram Protocol*, Protocolo de Datagramas de

	Usuario.
VM	*Virtual Machine*, Máquina Virtual.
VON	*Virtual Optical Networking*, Red Óptica Virtual.
WDM	*Dense Wavelength Multiplexing*, Multiplexación por División de Longitud de Onda.
WLAN	*Wireless Local Area Network*, Red Inalámbrica de Área Local.

CAPÍTULO 1

1. INTRODUCCIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

El crecimiento exponencial de las telecomunicaciones y el uso masivo de internet ha generado nuevos conceptos y experiencias, dando lugar a nuevas tecnologías, como son las redes, redes definidas por software (SDN, *Software Defined Networking*) y conmutación de etiquetas multiprotocolo generalizado (GMPLS, *General Multiprotocol Label Switching*). De manera general, el concepto de SDN hace referencia a un conjunto de técnicas relacionadas con el área de redes computacionales, cuyo objetivo es facilitar la implementación y diseño de servicios de red de una manera determinística, dinámica y escalable, evitando al administrador de la red gestionar dichos servicios a bajo nivel, mediante la separación del plano de control (*software*) del plano de datos (*hardware*) [1].

Las redes ópticas elásticas (EON, *Elastic Optical Networks*) se constituyen en la mejor alternativa para solucionar los inconvenientes encontrados en las redes basadas en distribución estática del espectro óptico, como es el caso de las redes ópticas basadas en multiplexación por división de onda (WDM, *Wavelength Division Multiplexing*) con rejilla fija. Además, se generalizó el uso de GMPLS en redes ópticas enfocado al plano de control, puesto que las capas restantes pueden usar físicamente diferentes tipos de datos con la intención de cubrir tanto la señalización como la parte de enrutamiento en el plano de control [2].

Estas tecnologías representan una tendencia en las telecomunicaciones dado que buscan incrementar eficiencia y adaptabilidad, disminuyendo costos, adoptando SDN como la tecnología clave para la virtualización de sus redes,

mejorando su rendimiento, reduciendo la complejidad de la red y controlando el despliegue de recursos de infraestructura, evitando que la introducción de aplicaciones como BigData se convierta en un cuello de botella en las redes tradicionales [3].

Debido al alto crecimiento en la demanda de ancho de banda por los nuevos servicios de telecomunicaciones se ha generado un interés masivo en las redes ópticas durante los últimos años. La principal razón de este crecimiento ha sido el aumento exponencial del número de usuarios y el incremento del número de aplicaciones de internet, superando al tradicional tráfico de voz, aumentando el interés en el protocolo de internet (IP, *Internet Protocol*)[4].

Sin embargo, al momento de brindar solución al problema clásico de Enrutamiento y Asignación de Espectro Óptico (RSA, *Routing and Spectrum Allocation*) se han propuesto diferentes soluciones que tratan de demostrar la eficiencia de los algoritmos desarrollados en el desempeño de las redes EON. Una de estas soluciones se enfoca hacia el empleo de GMPLS y/o elementos de cálculo de rutas (PCE, *Path Computation Element*) como mecanismo de enrutamiento basado en el protocolo de conmutación de etiquetas; Otra solución se enfoca en los mecanismos basados en topologías virtuales, su práctica podría ayudar directamente en la solución de RSA y a una posible mejora en el desempeño de una SDEON.

Teniendo en cuenta los elementos mencionados, en el presente trabajo de grado se propone el desarrollo de un mecanismo de enrutamiento basado en procesos de topologías virtuales como una posible solución a RSA y se la compara con el mecanismo clásico basado en GMPLS/PCE.

1.2. ORGANIZACIÓN DEL TRABAJO DE GRADO

En el Capítulo 2, se abordarán conceptos básicos y temáticos relacionados con MPLS, GMPLS/PCE, Topologías Virtuales y SDEON definiendo generalidades de las redes ópticas y técnicas de conmutación óptica buscando ofrecer al lector una base de conceptos que se tratarán durante el desarrollo del trabajo de investigación.

En el Capítulo 3, se describen y comparan las herramientas software seleccionadas para el desarrollo del trabajo de grado resaltando OMNeT++ como el software definitivo para el desarrollo del trabajo de grado. Se incluye la metodología usada para el desarrollo del trabajo de grado que es el modelo en cascada debido a que cada proceso es muy predecible permite visualizar de manera más fácil la evolución del proyecto, de la misma manera se define la metodología de simulación utilizada que consiste en una serie de pasos donde se sigue una metodología de enfoque practico lo que permite evaluar el desempeño de la red en cada escenario de prueba.

En el Capítulo 4, se hará énfasis en el diseño e implementación del algoritmo de enrutamiento dinámico basado en Topologías Virtuales propuesto para la evaluación de su desempeño en redes SDEON, además se presentan los escenarios de simulación y pruebas, para la evaluación del desempeño de las redes ópticas basadas en GMPLS/PCE y SDEON, analizando la pérdida de paquetes, la probabilidad de bloqueo y los retardos que experimentan los paquetes al cruzar por la red.

En el Capítulo 5, se incluyen las conclusiones con base al análisis de los resultados obtenidos en la ejecución de las simulaciones, descritas en el capítulo anterior. Adicionalmente, se incluyen posibles trabajos futuros, extendiendo así esta línea de investigación.

CAPÍTULO 2

2. GENERALIDADES

En el presente capítulo se expone una base conceptual sobre las redes de transporte de datos y protocolos de conmutación cuyos elementos básicos sirven de guía para desarrollar los algoritmos propuestos, igualmente, en principio se aborda MPLS, además, se requiere información sobre redes ópticas, como es el caso de las redes de fibra óptica y redes centralizadas definidas por software, en la cual se relaciona definiciones teóricas, componentes, características, arquitecturas, entre otros con la finalidad de comprender las tecnologías a utilizar y construir una red inicial de prueba.

2.1. MultiProtocol Label Switching

La Conmutación de etiquetas multiprotocolo (MPLS, *MultiProtocol Label Switching*) es un mecanismo de transporte de datos estándar creado por el Grupo de trabajo de ingeniería de Internet (IETF, *Internet Engineering Task Force*) y definido en el RFC 3031[1], que opera entre la capa de enlace y la capa de red del modelo OSI, diseñado para unificar el servicio de transporte de datos para las redes basadas en circuitos y paquetes, incluyendo tráfico de voz y tráfico de paquetes IP[5].

2.1.1. Arquitectura

- **Etiqueta:** es un campo de 20 bits que establece una relación entre el tráfico y la Clase de Equivalencia de Reenvío (FEC*, Forwarding Equivalence Class*) específica. Esta etiqueta es transportada en la cabecera MPLS de un paquete e identifica el camino por el que debe ser enviado, en donde la asignación de dicha cabecera se realiza en función

[1] Arquitectura de conmutación de etiquetas multiprotocolo, con el objetivo de proporcionar características de redes orientadas a conexión a redes no orientadas a conexión

de la dirección de destino, el tipo de servicio, la pertenencia a una red privada VPN y/o siguiendo otros criterios.

- **Cabecera MPLS:** La cabecera genérica MPLS es un campo de 32 bits que se añade a un paquete, entre las cabeceras de nivel 2 y 3, y que define una serie de características y requisitos para su transmisión en una red MPLS. La estructura de la cabecera se muestra en la figura 1.

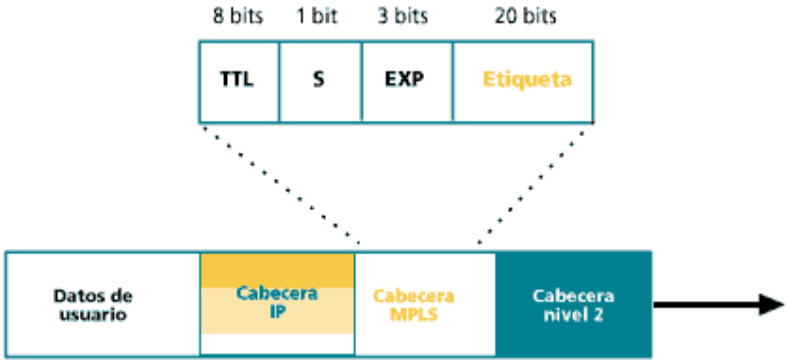

Figura 1 Estructura de la cabecera MPSL, tomada de [31].

- **EXP:** Campo que permite la diferenciación de distintos tipos de tráficos con el objetivo de mejorar el rendimiento de un tipo de tráfico con respecto a otros, este campo se conocía como Clase de servicio (CoS, *Class of Service*).

- **Stack:** También nombrado como parte inferior de la pila (BoS, *Bottom of Stack*). Con este bit se soporta la jerarquización de etiquetas *(Label Stacking)*, si el valor es positivo indica que hay más etiquetas añadidas de lo contrario indica que es la última etiqueta de la jerarquía.

- **TTL:** Proporciona la funcionalidad de Tiempo de Vida (TTL, *Time to Life*) del paquete típica de IP, la cual permite mitigar el efecto de posibles bucles en la red decrementando el valor inicial en una unidad por cada salto o nodo por el que pase el paquete.

2.1.2. Label Switch Router

Un enrutador de conmutación de etiquetas (LSR, *Label Switch Router*) es un router capaz de recibir, procesar y enviar paquetes con etiquetas MPLS por un enlace de transmisión [6].

En una red MPLS existen dos tipos de LSR:

- Enrutador de Etiquetas de Borde (LER, *Label Edge Router*): situados en los extremos de la red MPLS, siendo estas pasarelas con las redes tradicionales (Ethernet, Frame Relay,ATM...).

- Intermedio LSR: Situados dentro de la red MPLS, reciben y transmiten los paquetes etiquetados por los enlaces correspondientes.

Un LSR es capaz de realizar tres operaciones básicas: añadir (*Push*), eliminar (*pop*) e intercambiar (*swap*) etiquetas MPLS, en donde la acción de añadir la primera etiqueta en un paquete se denomina imposición y la acción de eliminar la última etiqueta de un paquete de denomina disposición como se muestra en la Figura 2.

2.1.3. Label Switched Path

Un Camino Conmutado de Etiquetas (LSP, *Label Switched Path*) es el camino de tráfico específico a través de la red MPLS, resaltando que son unidireccionales por definición. En el extremo inicial se sitúa el *Ingress* LSR, el cual clasifica y etiqueta los paquetes, decidiendo qué paquete pertenece a cada FEC. En el extremo final se sitúa el *Egress* LSR, encargado de eliminar la etiqueta correspondiente a ese LSP. Cabe resaltar que no necesariamente el Ingress LSR de un LSP tiene que ser un LER para etiquetar el paquete, en este caso se añadirá una nueva etiqueta a la pila de los LSPs anidados y esto es gracias al Label Stacking, este paso se denomina como jerarquización de LSPs.

2.1.4. Label Information Base

La Base de Información de Etiquetas (LIB, Label Information Base) es la tabla de enrutamiento de un LSR, en donde cada entrada de la tabla contiene interfaz de entrada- etiqueta de entrada y su correspondiente etiqueta de salida- interfaz de salida. También se conoce como la Base de Información de Envío de Etiqueta (LFIB, Label Forwarding Information Base), como se muestra en la figura 2.

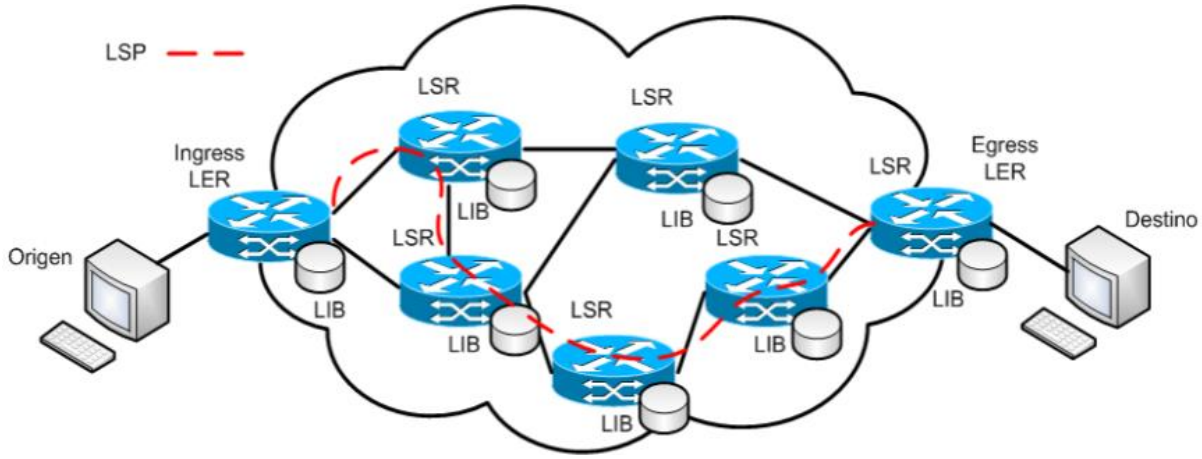

Figura 2 Esquema de una red MPLS, tomada de [32].

2.2. GMPLS

GMPLS es una extensión de MPLS, cuyo objetivo es proporcionar características de redes orientadas a conexión, soportando múltiples tipos de conmutación. A diferencia de la arquitectura MPLS que fue diseñada para soportar la transmisión de datos basadas en etiquetas, en la arquitectura GMPLS los routers de conmutación por etiquetas (LSRs, *Label Switching Routers*) son capaces de reconocer los límites de las celdas o paquetes y de procesar sus cabeceras. Así mismo entre estos routers se intercambia información sobre la topología de la red y se construyen tablas de enrutamiento. Esta arquitectura está siendo ampliada para incluir LSRs con dispositivos que permiten la conmutación basada en intervalos de tiempo (*Time Slots*), longitudes de onda o puertos físicos [7][8][9].

En GMPLS existe una separación entre el plano de datos y el plano de control que puede ser realizada de manera lógica o física, en donde una separación lógica los tráficos de datos y de control viajan sobre la misma red. Por otra parte, en una separación física significa que el control de la red de datos se realiza a través de otra red externa, como por ejemplo una red de datos óptica con un plano de control sobre una red IP tradicional.

2.2.1. Etiqueta Generalizada

Este nuevo formato de etiqueta puede representar un paquete, una celda, un *Time Slot* o una longitud de onda, lo que permite que los Routers soporten diferentes tipos de conmutación, por lo tanto, la longitud de la etiqueta generalizada, así como su formato y contenido dependen del tipo de conmutación del enlace.

2.2.2. LSP Bidireccional

En GMPLS todos los LSPs son bidireccionales, por lo tanto, ambos sentidos deben tener las mismas características y parámetros de ingeniería de tráfico[2], siendo esto útil en servicios como videoconferencia o VoIP.

2.2.3. Adyacencia de reenvío

Permite crear LSPs de nivel superior para realizar cambios de conmutación intermedios como si fuese un enlace virtual con sus propias características de ingeniería de tráfico, denominando a este enlace como FA-LSP[3].

[2] Proceso de adaptar los flujos de tráfico a los recursos físicos de la red para optimizar su utilización, de manera que no haya alguno sobrecargado (cuellos de botella) mientras otros están infrautilizados.

[3] Constituido por dos LSRs extremos, actuando como si fuese un enlace más de la red para establecer conexión a otros LSPs superiores, sin preocuparse de la gestión del mismo.

2.2.4. Link Bundling

En una red óptica se despliegan decenas de fibras paralelas entre nodos, en donde cada una de ellas transporta centenares de longitudes de onda. El *Link Bundling* permite agrupar todos estos enlaces con una misma señalización de manera que la información de control que se transmite entre los LSRs de la red disminuye considerablemente. Se debe tener en cuenta que todos los *Links Bundling* deben tener como inicio y final el mismo par de LSRs, el mismo tipo de conmutación y las mismas características de ingeniería de tráfico.

2.2.5. Administración de Red

Los proveedores de servicios requieren monitorear, configurar y controlar los recursos de red y para ello utilizan un sistema de administración de red (NMS, *Network Management Service*), combinando herramientas hardware y aplicativos softwares cuyos objetivos básicos son:

- **Operación:** Mantener el correcto funcionamiento de la red y monitorear para la detección de posibles fallas.

- **Administración:** Conocer los recursos de la red y asignarlos de manera adecuada.

- **Mantenimiento:** Repara y actualizar al presentar insistentes fallas.

- **Aprovisionamiento:** Configurar los recursos para ofrecer el servicio solicitado

La comunicación de los dispositivos de red y el NMS, se lleva a cabo mediante el protocolo SNMPv3 *(Simple Network Management Protocol)* definido por la IETF [RFC 3410].

2.3. PCE (*Path Computation Element*)

Elemento de cálculo de trayecto es un nodo de red que permite determinar y encontrar una ruta más adecuada para transmitir datos entre el origen y destino,

para la disminución de tráfico en redes MPLS y GMPLS, proporcionando la ruta más adecuada para cada LSP, etiqueta de ruta conmutada, que se configura en la red.

El cálculo de la ruta se realiza previamente en un sistema de gestión o en la cabecera de cada LSP, capaz de calcular rutas para un solo o conjunto de servicios como se observa en la figura 3.

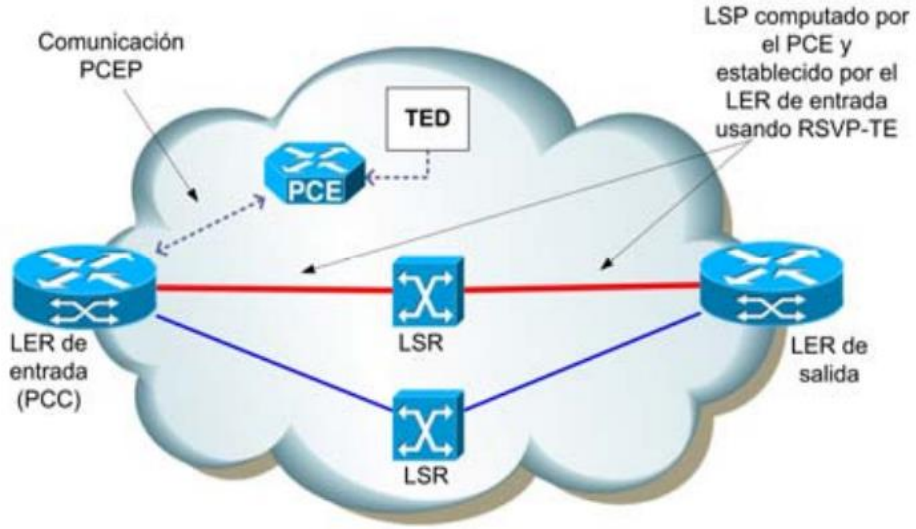

Figura 3 PCE, tomada de [33].

Existen tres elementos importantes en la arquitectura básica PCE:

- PCE (*Path Computation Element*): es el encargado de calcular la mejor ruta para el envío del paquete

- PCC *(Path Computation Client)*: es el cliente que solicitará al PCE el cómputo de caminos.

- PCEP *(Path Computation Element Communication Protocol)*: es el protocolo de comunicaciones mediante el cual se comunican PCC Y PCE.

En algunas redes, el Enrutador de borde de etiqueta (LER, L*abel Edge Router*) permiten la entrada a redes MPLS actuando como PCC, solicitando al PCE que compute un LSP desde el hasta el LER de salida del dominio, utilizando el protocolo PCEP lo que proporciona la suficiente funcionalidad para permitir al

LER/PCC de entrada realizar la solicitud de cómputo, calculando la mejor ruta basándose en la información contenida en la base de datos de ingeniería de tráfico (TED, *Traffic Engineering Database*), base de datos que contiene el grafo de estado de la red e información adicional que pueda ser de gran utilidad. Una vez calculada la ruta, el PCE usará de nuevo PCEP para enviar la respuesta al LER/PCC que realizó la solicitud [10][11].

2.4. Topologías Virtuales

La virtualización de redes busca eludir la rigidez de internet, permitiendo que múltiples redes virtuales, independiente de su topología y configuración, compartan una única red física. Este concepto inicialmente fue aplicado a los protocolos de capas superiores, pero estudios recientes han logrado virtualización desde la capa física, que tradicionalmente se basan en la división de longitudes de ondas (WDM, *Wavelength Division Multiplexing*) [12][13].

Las redes virtuales ópticas (VON, *Virtual Optical Network)* buscan manejar las crecientes demandas de tráfico de una manera más eficiente, lo que ha implicado buscar nuevas formas de trabajo en las mismas, por ejemplo la multiplexación óptica por división de frecuencia ortogonal (OOFDM, *Optical Orthogonal Frequency Multiplexing*) se ha propuesto como una tecnología viable para transmisión óptica, en el que permite que el ancho de banda de la fibra pueda dividirse en subportadoras mucho más finas con longitudes de onda mucho más cortas permitiendo la variedad de esquemas de modulación y que las bandas subportadoras puedan asignarse a un servicio según lo necesario. De esta manera se disminuyen el uso insuficiente de los recursos de la red, ajustándose a los requisitos del servicio a ofrecer de una manera mucho más flexibles que en redes basadas en WDM, es de aquí que nace el concepto de las redes ópticas elásticas (EON, *Elastic Optical Networks*).

El aprovisionamiento de segmentos de cada solicitud requiere de una parte de la red física, incluyendo longitudes de onda, dispositivos ópticos y/o conversores óptico-eléctrico-óptico (OEO) en los nodos regeneradores. Este modelo se puede clasificar además de dos formas:

- **Aprovisionamiento de sectores:** modelo específico en el que la solicitud de segmento incluye, longitudes de onda específicas de

cada enlace físico, puertos específicos de cada dispositivo óptico y el número de circuitos OEO en cada regenerador.

- **Aprovisionamiento *slicing*:** modelo flexible en donde la solicitud de segmento incluye el requisito de velocidad de bits en cada enlace físico, el número de puertos en cada dispositivo óptico y la accesibilidad de cada par de nodos. El proveedor asigna longitudes de onda en cada enlace, los puertos en cada nodo y los conversores OEO en cada nodo regenerador a la solicitud.

En VN Mapping(*Mapeo en redes virtuales*), cada solicitud es una topología de red virtual que incluye nodos virtuales y/o enlaces virtuales. El proveedor asigna nodos físicos a nodos virtuales y asigna longitudes de onda en cada enlace a los enlaces virtuales.

2.5. Redes ópticas elásticas o Flexigrid

Las redes ópticas elásticas, como se muestra en la figura 4, son redes adaptativas, flexibles y eficientes en el espectro basadas en OFDM proporcionando una técnica de modulación de portadora única, debido a que el flujo de datos es multiplexado en múltiples subportadoras consecutivas de baja velocidad, aumentando la duración del símbolo y proporcionando una velocidad de datos mucho más alta.

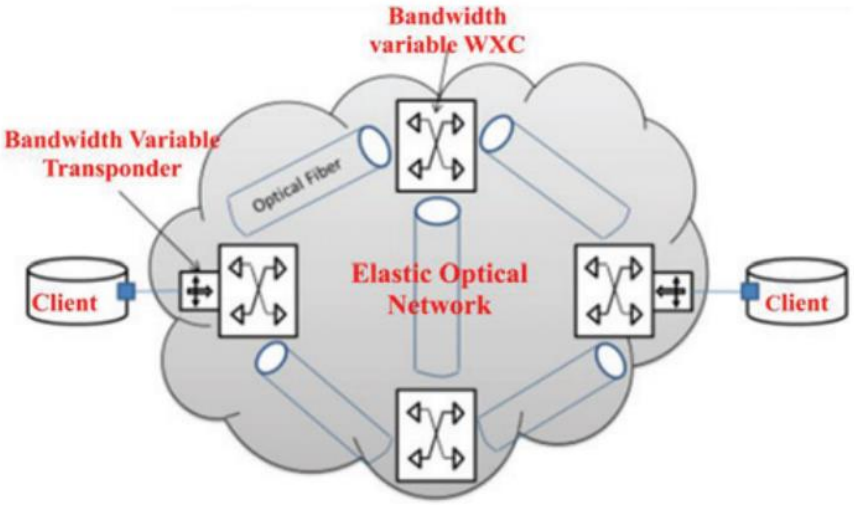

Figura 4 Arquitectura EON, tomada de [34].

Como se puede observar en la figura 4 de la arquitectura de una red óptica elástica o Flexigrid, el cliente se conecta a la red por medio de un transpondedor de ancho de banda variable para la conversión del dominio eléctrico a óptico, para así llevar la conexión por fibra gracias al conmutador de ancho de banda variable que enruta las solicitudes de los paquetes a través de la red [14][15].

La separación entre canales logra una mejor eficiencia espectral y obtener una red dinámica, buscando superar la barrera de los 400 Gbps. Las redes ópticas elásticas se componen de conmutadores ópticos, *WXCs*, en la red central y transpondedores de ancho de banda variable, BVT, en el borde de la red basados en multiplexación por división de frecuencias ortogonales.

Para obtener una red EON la rejilla de espectro debe de ser una rejilla flexible, establecida por la ITU-T en el estándar G-694.1, en la cual el espectro óptico de la ventana es dividido en pequeñas ranuras de frecuencia. En la actualidad,las redes EON cuentan con arquitectura SDN, las cuales se pueden aprovisionar y reconfigurar de manera flexible y dinámica, además de permitir la separación del plano de control del plano de datos y convierte la forma de control en una centralizada y flexible (SDEON, *Software Defined Elastic Optical network*).

En una EON, el plano de control es la tecnología clave para soportar el aprovisionamiento dinámico, flexible e inteligente en enlaces extremo a extremo, la detección de fallas y la recuperación de información. Son dos los planos

principales estudiados para EON, el primero se basa en el elemento de cálculo de ruta/conmutación de etiquetas multiprotocolo generalizado (GMPLS/PCE) en el que el RSA se calcula mediante el elemento de cálculo de ruta (PCE), siendo una tecnología muy útil para redes ópticas. Sin embargo, un plano de control basado en GMPLS/PCE aún no es la solución ideal para EON debido a su complejidad y al control distribuido, especialmente en situaciones de múltiples capas[16][15].

La segunda opción para el plano de control es SDN basado en OpenFlow, que permite el control centralizado de redes mediante programación. Esta arquitectura podría usarse en redes ópticas para obtener mayor flexibilidad e inteligencia para rutas con más de 3 saltos, el plano de control SDN tendrá menos latencia de aprovisionamiento de ruta que el plano de control basado en GMPLS/PCE debido al control centralizado en SDN, a partir de estos conceptos nace la red óptica elástica controlada por un controlador SDN (SDEON).

2.5.1. Arquitectura de red

La arquitectura SDEON tiene las mismas tres capas que la de SDN, con algunas diferencias. Se comienza con los elementos de red en el plano de datos de SDEON, son diferentes, los elementos de red en EON son transpondedores de velocidad de datos/ ancho de banda variable (BV) y WXC de ancho de banda variables(BV), a diferencia de los conmutadores tradicionales de una red IP. Otra diferencia está en los recursos del plano de control que en una SDEON son ranuras de espectro y por último el plano de control considera unas restricciones para el cálculo de enrutamiento y formatos de modulación y del espectro como se aprecia en la figura 5.

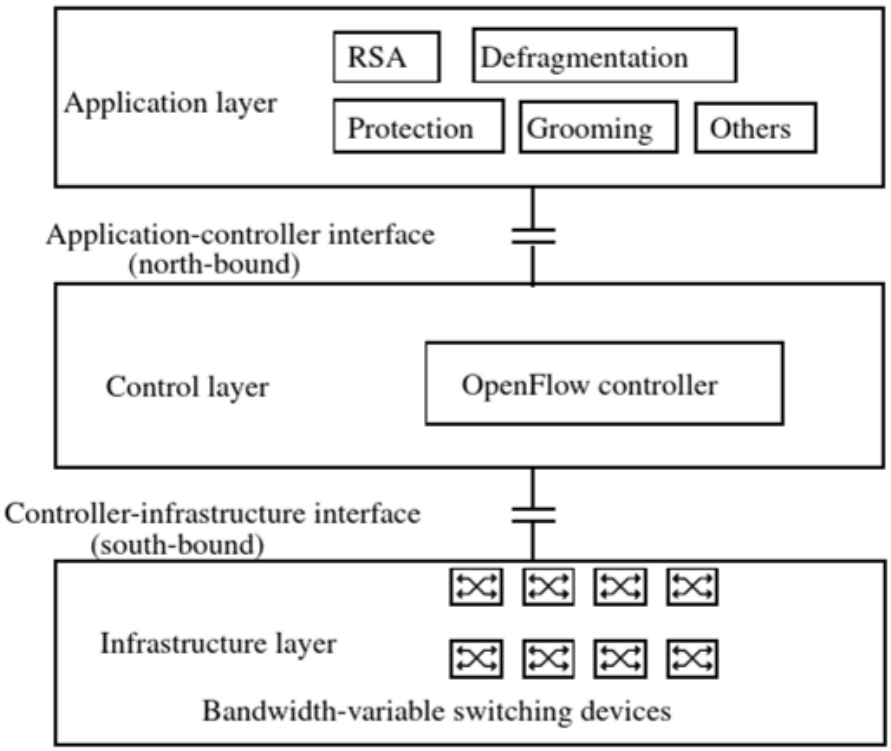

Figura 5 Arquitectura SDEON, tomada de [35].

2.6. Enrutamiento y Asignación de Espectro

RSA es el proceso que permite identificar la ruta adecuada para la conexión entre origen y destino y la asignación de ranuras de espectro adecuadas a la solicitud. RSA permite el ahorro de recursos espectrales, mejorando la operatividad de la red óptica en donde el requisito de capacidad de cada solicitud de conexión es caracterizado por una serie de ranuras de subportadoras [17][18].

Si se establecen trayectos de luz para cada conexión o solicitud, seleccionando la mejor ruta y asignando a ella la longitud requerida se conoce como enrutamiento y asignación de longitudes de onda (RWA, *Routing and Wavelength Allocation*). En las redes basadas en WDM de rejilla fija, las ranuras de espectro asignadas deben colocarse cerca una de la otra para lograr satisfacer la restricción de contigüidad del espectro, de no haber suficientes ranuras contiguas disponibles a lo largo de la ruta la conexión se puede dividir en múltiples y pequeñas demandas.

RSA tiene como objetivo principal encontrar una ruta con suficiente espectro libre para brindar el ancho de banda requerido por las demandas de tráfico. La asignación de espectro (SA, *Spectrum Allocation*) de una conexión óptica consiste en encontrar un canal que cumpla con las limitaciones de contigüidad y continuidad del espectro, para ello las ranuras de un canal deben estar cerca de la otra mientras que el canal asignado debe usar la misma frecuencia central para todos los enlaces que conforman esa conexión óptica [19].

RSA puede tener un enfoque en el que los algoritmos definen la ruta y el espectro contiguo disponible simultáneamente o se hace un enfoque de dos pasos el enrutamiento y la asignación de espectro que se resuelven secuencialmente, los siguientes son los aspectos a tener en cuenta [18]:

- **Contigüidad del espectro:** todos los slots de frecuencia del canal en una conexión óptica deben ser adyacentes en el espectro.

- **Continuidad del espectro:** es una conexión óptica para un enlace óptico desde el nodo origen al destino se debe asignar el mismo canal, es decir la misma frecuencia central.

- **No superposición del espectro:** un slot de frecuencia en un enlace solo se puede utilizar para una conexión a la vez.

- **Espectro limitado:** hay un número finito de slots en cada enlace.

- **Bandas de guarda:** es necesario separar las conexiones espectralmente adyacentes en cada fibra, por lo menos un slot no asignado para evitar interferencias.

2.7. Algoritmos RSA

RSA se divide en dos partes, como primera instancia la operación de enrutamiento que es el cálculo de la ruta entre el nodo origen y destino por medio de la topología de la red, y por otra parte se tiene la selección de los recursos

espectrales que se asignan a la petición y a la asignación de espectro definidos por una frecuencia central y un ancho de banda o slot.

2.7.1 Algoritmos RSA dinámicos (*Online*)

Consiste en enviar peticiones dinámicamente según su orden de llegada. La red se va ocupando conforme las peticiones que se van atendiendo, siendo este el problema más complejo debido a la entrada/salida de tráfico aleatorio y la fluctuación de las demandas de tráfico a lo largo del tiempo. Dependiendo del estado de la red, los recursos espectrales disponibles pueden o no ser suficientes para establecer una conexión. El estado de la red consta de los enlaces físicos y la asignación de espectro para todas las conexiones activas [20][21].

En este capítulo se ha hecho mención de los principales conceptos que se han tomado como base para la realización del trabajo de investigación, cada aspecto realiza su aporte al desarrollo del mismo.

CAPÍTULO 3

3. HERRAMIENTAS Y METODOLOGÍA DE TRABAJO

En este capítulo se describirán algunas de las herramientas de simulación más empleadas en diferentes trabajos de investigación con temáticas similares, finalmente se hará una elección de una de ellas basado en los objetivos y/o requerimientos del trabajo de grado a desarrollar. Posteriormente se menciona la metodología de trabajo seleccionada para el trabajo de investigación y su ejecución en el mismo. Finalmente, se realiza la descripción del modelo de red implementado y los componentes de red utilizados en dicha red.

3.1. Herramientas de simulación para redes ópticas

3.1.1. OMNET++

Simulador de eventos discretos de redes orientados a objetos, usado para modelar el tráfico de redes de telecomunicaciones, sistemas multiprocesadores distribuidos, protocolos y en general cualquier sistema que se pueda simular con eventos discretos[22][23].

El software provee un núcleo de librerías e interfaces de usuarios que se utilizan para la construcción de modelos y ejecución de simulaciones. Un modelo OMNeT + + consiste en la descripción de la topología del modelo en el lenguaje NED, definición de mensajes y código de módulos simples, con una estructura modular.

Los pasos para la construcción de un modelo de simulación OMNeT++ son:

- Descripción de la estructura, modelos y relaciones del sistema mediante lenguaje con representación gráfica.

- Implementación de los módulos simples en C++.

- Generar modelo, donde se compilan los módulos y se enlazan con la biblioteca de simulación.

- La configuración de la simulación en donde se especifican los parámetros adecuados para la ejecución de la simulación

Los módulos contienen algoritmos como funciones en el lenguaje C++, permitiendo manejar los eventos y definir los estilos de procesos haciendo uso de los diferentes conceptos de la programación orientada a objetos.

La topología de un modelo es definida en el lenguaje NED, facilitando la descripción modular de una red, así puede estar compuesta por la descripción de varios componentes para ser reutilizados en la descripción de otra red. Los archivos NED se pueden realizar mediante un editor de textos o el editor gráfico incorporado, permitiendo crear, programar, configurar y simular redes de comunicaciones sin necesidad de codificar en lenguaje NED, haciendo uso de diseños gráficos[22][23].

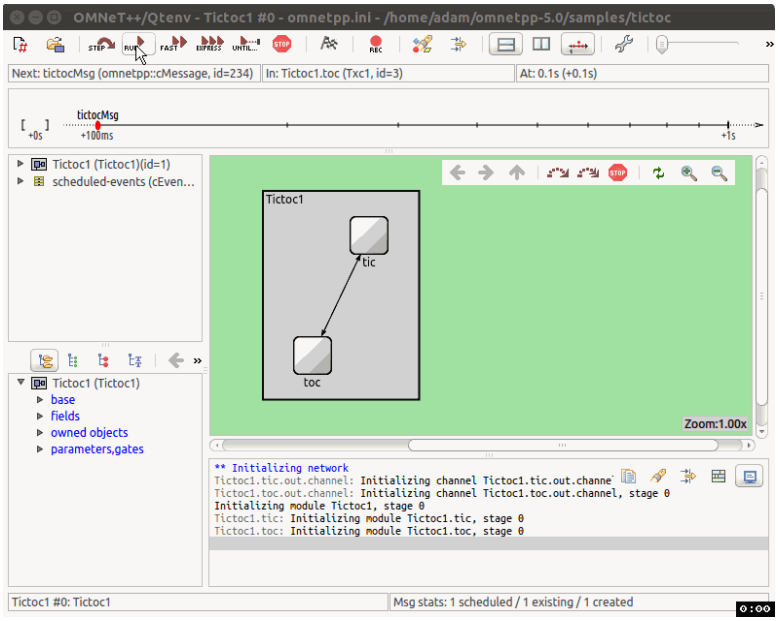

Figura 6 Herramienta de simulación OMNeT++

3.1.2. OptSim

Herramienta de software que soporta el diseño y la simulación de sistemas de comunicación óptica con técnicas de simulación de última generación, una interfaz gráfica de alto nivel, brindando instrumentos de medición similares a los de un laboratorio como se observa en la figura 7. El software ha estado disponible comercialmente desde 1998, lo que implica el uso de licencia para su operación y lo utilizan ingenieros líderes en organizaciones académicas e industriales de todo el mundo[24] [25] .

Cuenta con varios beneficios para los usuarios, como la creación de prototipos virtuales de sistemas de comunicación óptica, optimización de diseños para mejorar rendimientos y disminución de costos, Interconexión e interfaces con herramientas externas como Matlab, ampliamente usada en el campo educativo para el diseño de Sistemas de comunicación óptica coherentes, como PM-QPSK, PM-BPSK, PM-QAM, OFDM entre otros[24].

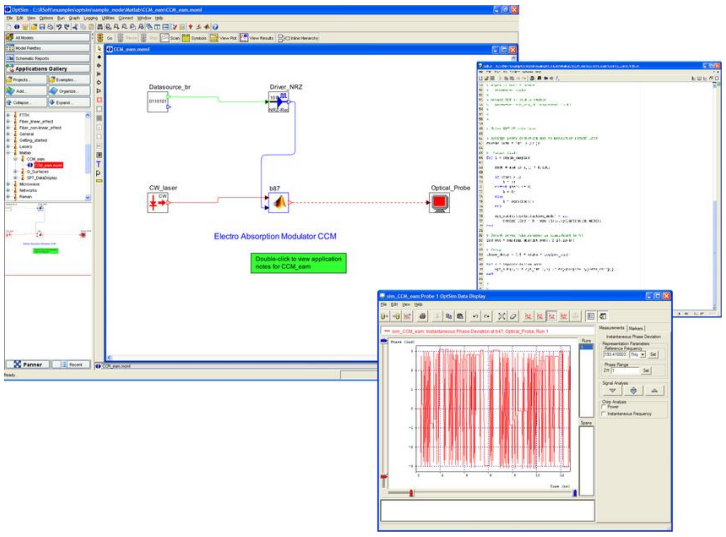

Figura 7 Interfaz de herramienta OptSim.

3.1.3. Open Network Operating System (ONOS®)

Es un controlador SDN de código abierto, es decir, no requiere licencia para su implementación, diseñado para crear soluciones en la Virtualización de

Funciones de Red (NFV, *Network Function Virtualization*) sobre SDN de próxima generación, brindando flexibilidad para crear e implementar nuevos servicios de red dinámicos con interfaces de programación simplificadas. ONOS, cuyo logo se observa en la figura 8, admite tanto la configuración como el control en tiempo real de la red, lo que elimina la necesidad de ejecutar protocolos de control de conmutación y enrutamiento dentro de la estructura de la red, es un software dinámico que gestiona distintas capas del modelo OSI[26] [27].

ONOS brinda la implementación de distintos componentes de red como switches y/o aplicaciones software para proveer servicios a los host destino o finales a redes adyacentes, y lograr interconexión entre diferentes clusters. Dentro de las redes de comunicaciones simuladas en ONOS, podemos encontrar funcionalidades como APIs, asignación de recursos, CLI, GUI, permisos, entre otros, ya que ONOS puede ser usado como sistema operativo donde el administrador puede desarrollar aplicaciones de monitorización SDN, permitiendo la exportación de métricas e información sobre la red[26] [27].

Figura 8 Logo de la herramienta ONOS.

En la tabla 1 se presenta una comparativa de las herramientas expuestas anteriormente.

CARACTERÍSTICAS	**OMNeT ++**	**ONOS**	**Optsim**
LICENCIA	Gratuita	Gratuita	Comercial
INTERFAZ	Media	Alta	Alta
PLATAFORMA	Windows, unix	Linux, Ubuntu	Windows, unix.
HEURÍSTICA	Si	Si	Si

GRAFICACIÓN	Buena	Muy buena	Muy buena
SOPORTE NIVEL 2 y 3 (ENLACE Y RED)	Si	Si	Si
USO EN LA ACADEMIA	Alto	Alto	Alto
FLEXIBILIDAD	Alta	Media	Media

Tabla 1 Comparación de herramientas de simulación, tomada de [35][36]

Para el desarrollo del trabajo de grado, la simulación de la SDEON y la implementación de los algoritmos de enrutamiento dinámico se ha seleccionado la herramienta OMNeT++, debido a su licencia libre, incorporación de diferentes clases ya establecidas y la capacidad de crear y simular todos los componentes modulares de la red. La herramienta Optsim no fue seleccionada puesto que se enfoca en características físicas de las redes ópticas, y la herramienta ONOS se diferencia de OMNeT++ puesto que este software es un controlador y sus componentes de red se conectan a través de la herramienta Mininet.

3.2. Metodología para el desarrollo del trabajo de investigación

Para el desarrollo del trabajo de grado, se ha considerado hacer uso de la metodología de trabajo denominada modelo en cascada o ciclo de vida clásico, considerada adecuada para abordar el problema debido a sus fases y procesos claros, además de ser ampliamente usada para el desarrollo de soluciones software[29].

El modelo del ciclo de la vida está compuesto por 5 fases, las cuales suceden una tras otra a medida que los requisitos de cada una son completados,

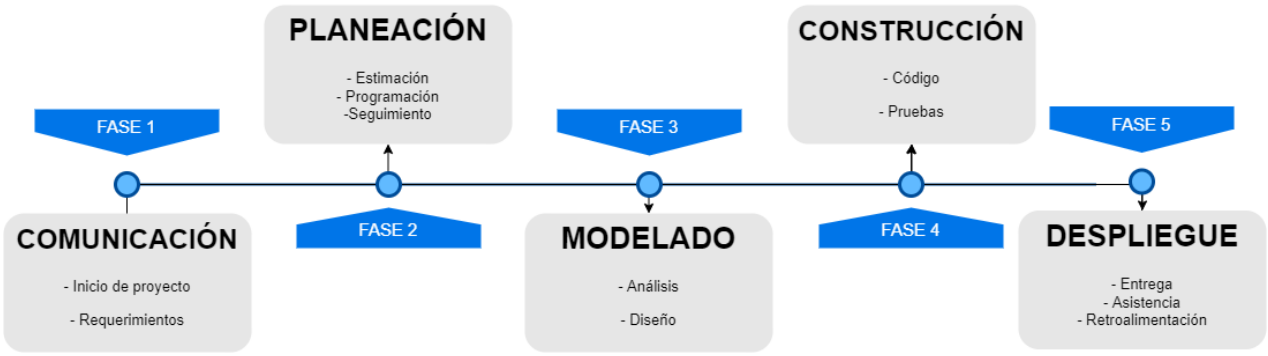

Figura 9 Modelo Cascada[29].

lo que permitirá avanzar de manera ordenada[28]. En la figura 9 se puede observar el proceso de vida del modelo en cascada.

A continuación, se describirán brevemente cada una de las fases, en su respectivo orden[29]:

- **Fase de recolección de información o comunicación:** En esta fase se busca entender las necesidades finales del cliente y se levantan todos los requisitos funcionales del proyecto de forma clara, de modo que se conozca desde un comienzo que debe realizar el software una vez terminado el proyecto.

- **Fase de planeación:** En esta fase se descompone el proyecto en partes más pequeñas que pueden ser desarrollados por separado o de forma más sencilla.

- **Fase de modelado:** En esta fase se diseñan los algoritmos que darán la solución al problema de usuario

- **Fase de construcción:** En esta parte se implementa el código desarrollado en la herramienta seleccionada y se hacen pruebas de funcionamiento del software obtenido.

- **Fase de despliegue:** Fase final, en que se entrega el proyecto y se hace un constante monitoreo del mismo para verificar que cumpla los requisitos de usuario. Se crea un canal de retroalimentación para corregir posibles errores del software.

3.3. Metodología de simulación

Para el estudio de la evaluación del desempeño de los modelos de red se sigue una metodología de enfoque práctico como se observa en la figura 10, que contiene una serie de pasos, de lo que se puede considerar una metodología de desarrollo de simulación a seguir, algunas de sus frases se pueden ajustar a lo requerido y es susceptible a cambios [29].

Se desarrollan las fases propuestas dentro de la metodología para la obtención de los modelos de desempeño y simulación, como se detallan a continuación:

- **Definición del escenario de prueba:** se busca realizar un análisis del desempeño de una red SDEON basado en topologías virtuales, implementando un algoritmo de enrutamiento, teniendo en cuenta aspectos teóricos que se investigan de diferentes artículos o trabajos de grado los cuales se basan en los objetivos planteados en el trabajo de grado, en este caso, se comienza con una red genérica basada en la topología NSFNeT con enfoque centralizado (red basada en SDN).

- **Modelo de red de prueba:** se plantea un modelo de red de prueba, esta puede corresponder a una red SDON inicial, es decir una red con características SDN (centralizada) y con enfoque hacia redes ópticas, basada en un método de búsqueda de rutas y asignación de espectro, que por cada escenario propuesto se adecue a los requerimientos planteados buscando una posible solución a la pregunta de investigación que se desea resolver.

- **Diseño e implementación del algoritmo:** con base en la teoría investigada y el modelo de red de prueba se procede al diseño del algoritmo de enrutamiento basado en topologías virtuales para el análisis de su desempeño en comparación a métodos

convencionales que emplean las redes EON, en este caso se toma una SDEON como red de prueba.

- **Prueba en red sin algoritmo:** se realiza el proceso de simulación de la red SDEON sin el algoritmo basado en topologías virtuales.

- **Prueba red con algoritmo:** se realiza la simulación de la SDEON al implementar el algoritmo basado en topologías virtuales.

- **Comparación de los resultados:** se observa el desempeño de la red mediante el análisis de los parámetros de probabilidad de pérdida de paquetes y los retardos extremos a extremo, resultados obtenidos en los dos escenarios principales (sin y con la inclusión del algoritmo basado en topologías virtuales).

- **Conclusiones:** Basados en los resultados obtenidos se definen conclusiones que se derivan de las pruebas bajo las condiciones ya establecidas.

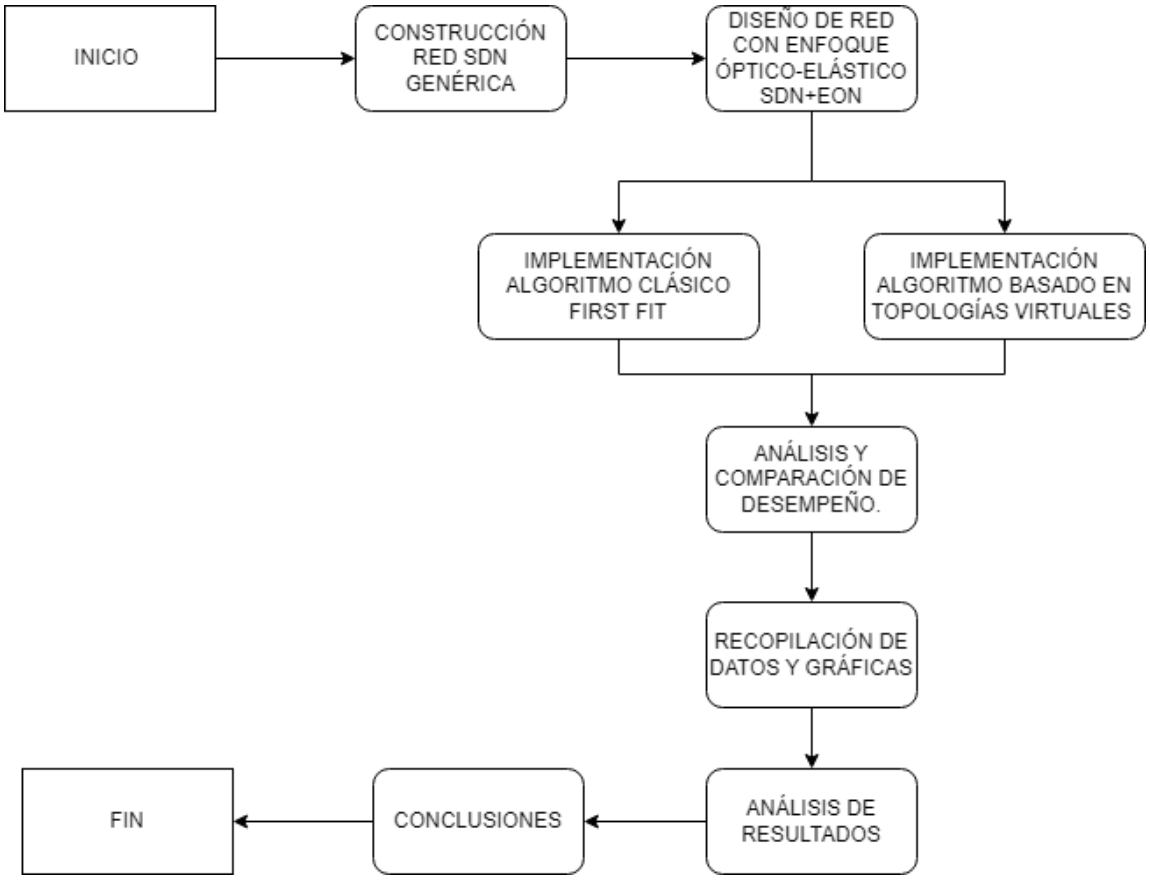

Figura 10 Metodología de simulación.

3.4. Definición del modelo de simulación.

El diseño de un modelo de simulación basado en la herramienta seleccionada, corresponde a una fase de gran importancia y definitiva para el desarrollo del trabajo investigativo, esencial para la elaboración del plan de pruebas y asegurar resultados coherentes, así mismo, define el modelo adecuado de la red y la interconexión de los elementos que la componen, brindando al lector una mejor comprensión del proceso realizado.

La red SDEON esta compuesta por elementos que deben ser definidos dentro del modelo de simulación, en este caso, son los nodos los elementos principales para la inicializar la operación en la red, lo que conlleva a su construcción a partir de módulos simples que los componen. Para la creación de

las peticiones y paquetes de datos se ha desarrollado un módulo denominado APP, para la transformación óptico-eléctrica o viceversa, se ha desarrollado un módulo denominado BVT y para conmutar los mensajes que ingresan a la red hacia el controlador se ha desarrollado un módulo denominado BWXC, los cuales son detallados a continuación.

3.4.1 Módulo simple APP

Encargado de realizar las peticiones a la red, solicitudes que llegan con un ancho de banda requerido, cumpliendo la función de usuario final por lo que puede representar un host, las peticiones se realizan siguiendo una distribución uniforme generando tráfico en la red.

Cada petición que realiza está formada por unos parámetros y el mensaje o paquete que lleva la petición, los lleva implícitos en su interior como se evidencia en la figura 11.

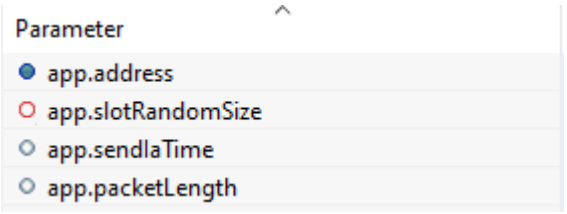

Figura 11 Parámetros del módulo app en la herramienta OMNeT++.

La petición generada inicia por con una dirección de destino, *address*, un número de slots, que son asignados por el módulo app que pueden ser aleatorios o fijos, seguido de s*endIaTime* que es el tiempo que tarda en producirse nuevas peticiones y entre más corto sea el tiempo aumentará la frecuencia de peticiones a llegar, siendo así el parámetro encargado de controlar el tráfico en la red.

Por último, está el *packetlengh*t que es la cantidad de información que tiene cada paquete a transportar, medida en bytes.

Como se muestra en la figura 12, el mensaje óptico generado en cada petición se encuentra conformado por una dirección de origen, una dirección

destino, el número de slots para la petición y un indicador de estado de mensaje para confirmar la solicitud cuando es enviado al controlador, adicionalmente cuenta con un parámetro de color para graficar en las ranuras de slots y diferenciar visualmente el estado de la petición (en tránsito o disponible).

```
int srcAddr;
int destAddr;
int slotReq;
int msgState;
```

Figura 12 Parámetros del paquete en la herramienta OMNet++.

3.4.2 Módulo simple BVT

Este módulo cumple la función de una interfaz óptico-eléctrica y viceversa según sea el caso de transmisión o recepción. Una vez las solicitudes son generadas en la app ingresan a este módulo encargado de transformar la interfaz a una tecnología óptica que permitirá la conexión con el siguiente módulo *BWXC*.

Este módulo representa la interfaz de red en una conexión punto a punto. Sus parámetros, como se muestran en la figura 13, hacen referencia a entradas, IN, y salidas, OUT, que llegan desde la fuente de peticiones o del módulo BWXC, según sea el caso, en donde las conexiones son ópticas mientras que las conexiones *localin* y *localout* con conexiones eléctricas.

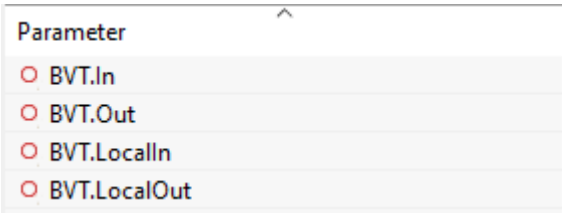

Figura 13 Parámetros del módulo BVT en la herramienta OMNet++.

3.4.3 Módulo simple BWXC

El módulo BWXC cumple la función de enrutador o conmutador óptico encargado de direccionar los mensajes que ingresan a la red, diseñado para el enrutamiento dinámico dentro de la red. Como se observa en la figura 14, el módulo BWXC tiene los siguientes parámetros:

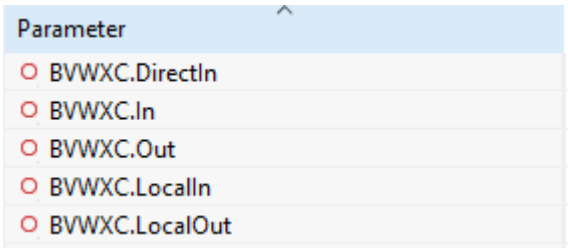

Figura 14 Parámetros del módulo BWXC en la herramienta OMNet++.

La entrada *directin* es el encargado de comunicar al módulo con el controlador, por medio de unas entradas especiales que brinda la herramienta OMNeT no requiere de una conexión física lo que hace que no consuma tiempo de ejecución dentro de la red, por medio de esta entrada se envía el mensaje óptico, si el estado del mensaje es LIGHTPATH_REQUEST es enviado al controlador de no ser así envía el paquete a la compuerta y al nodo que esté descrito en la tabla de enrutamiento, LIGHTPATH_ASSIGMENT.

Las entradas *in, out* son las conexiones entre los enrutadores y estas pueden variar según la capacidad de la red, y las entradas *localin* y *localout* son las conexiones eléctricas con el BVT.

3.4.4 Módulo Controlador

El controlador es el encargado de la gestión de la red y del manejo de sus recursos, la ruta y el espectro, con el fin de que la red sea centralizada y que cumpla con las características de una SDON Flexigrid. Fue necesario el diseño de módulos simples como se aprecia en la figura 15.

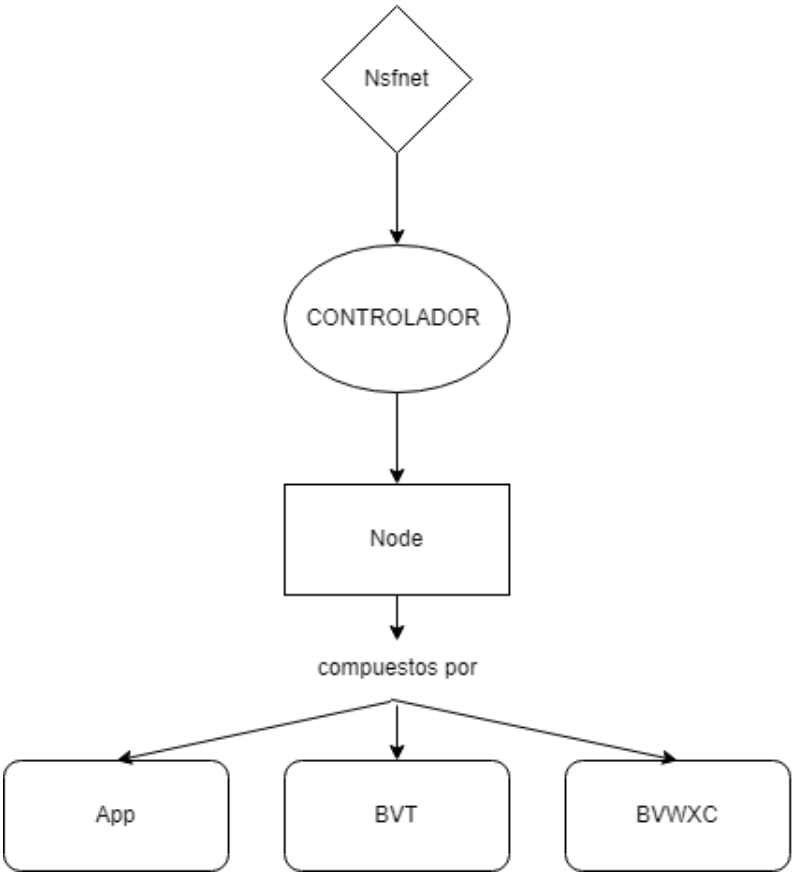

Figura 15 Diagrama del módulo compuesto Node.

El módulo controlador(*controller*) actúa según la cantidad de slots que requiera la petición y cantidad de slots disponible en el canal óptico, OMNeT++ ofrece unas diversas clases especiales para diseño de redes como es *cTopology* que trabaja con nodos y compuertas para representar las diferentes conexiones de la red, permitiendo calcular la ruta más cercana de un nodo destino. Como se observa en la figura 16, el módulo controlador tiene los siguientes parámetros.

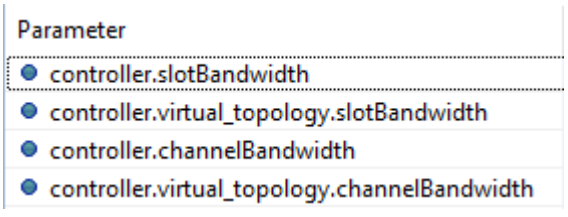

Figura 16 Parámetros del módulo controlador en la herramienta OMNet++.

El parámetro *slotbandwidth* es el ancho de banda que tiene cada slot de 12.5 GHz, basados en la recomendación de la ITU T G.694.1, y *channelandwidth* es la capacidad del canal o cantidad de slots disponibles.

Se implementa el método que calcule un número mayor de rutas entre nodos, debido a que la función anteriormente mencionada solo provee una ruta. apoyándose en la teoría de grafos logra implementar una estructura en C++ y así lograr calcular y almacenar las rutas necesarias para una sola conexión.

Todas las rutas entre los nodos fuente y destino se almacenan en un archivo texto llamado *routes*, como se muestra en la figura 17, guardando información de cada nodo y compuerta del enrutador por los cuales pasa cada ruta encontrada, el número de identificador del paquete, el color asignado por ruta, número de saltos de ruta y un identificador por cada ruta. Una vez el paquete llega al módulo controlador se calcula cada uno de los datos mencionados, se almacena y se usa posteriormente en futuras peticiones. Si una ruta entre nodos se encuentra almacenada el controlador busca la fuente y el destino y asigna el enrutamiento dinámico de acuerdo a los archivos almacenados y no realiza el cálculo de ruta nuevamente.

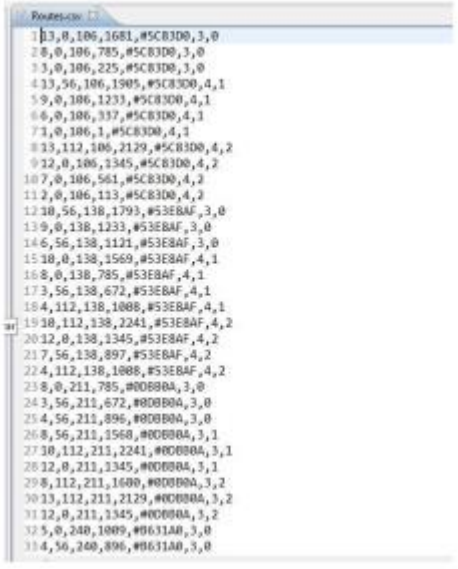

Figura 17 Archivo routes.

3.4.4.1 Módulo Simple Spectrum

Este módulo es el encargado de realizar la asignación de espectro a cada una de las conexiones de la red, al establecer la ruta para las solicitudes que provienen de los nodos, asigna el espectro que permite establecer la conexión. El módulo está compuesto por dos algoritmos que permiten la asignación del espectro y son el *FirstFit*, algoritmo de la EON, y el algoritmo propuesto de enrutamiento dinámico basado en topologías virtuales cumpliendo con las condiciones de continuidad y contigüidad de espectro, el conteo de paquetes perdidos y la probabilidad de bloqueo, información elemental para el análisis y evaluación del desempeño de la red.

3.4.4.2 Modulo Simple Topology

Este componente es el encargado de recibir las peticiones enviadas desde cada uno de los nodos de la red y así mismo asignarles una ruta de conexión aplicando el algoritmo de la ruta más cercana, permitiendo al controlador escanear la topología de la red y asignar la mejor ruta de conexión para el envío del paquete. Adicional este módulo escribe la tabla de enrutamiento dinámico para que los módulos *BVWXC* direccionen el paquete por la ruta indicada, el diagrama del controlador se muestra en la figura 18.

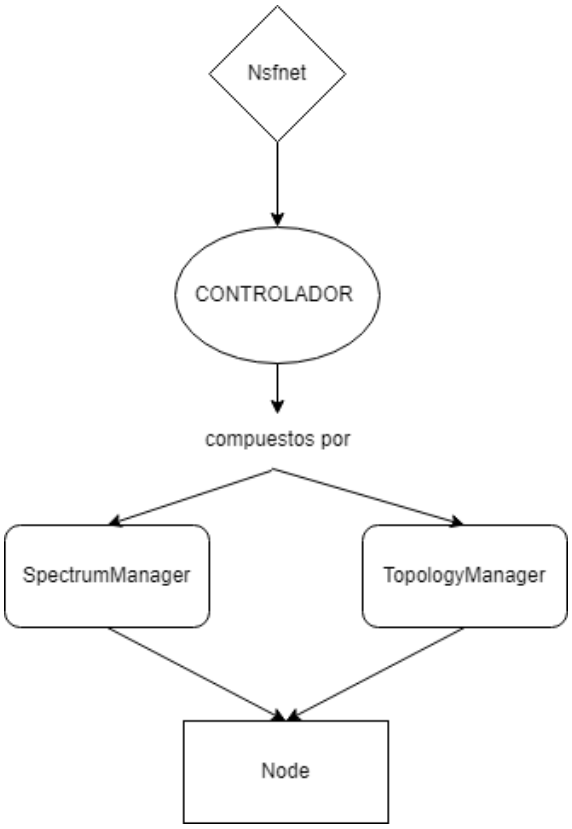

Figura 18 . Diagrama del controlador.

3.5 Caracterización de la red inicial

Una vez diseñados e implementados los módulos, descritos en la sección 3.4, se procede a la puesta en marcha del modelo de red óptica basada en la topología irregular NSFNeT, la cual consta de catorce nodos distribuidos en diferentes ubicaciones geográficas de los Estados Unidos de América interconectados a través de 42 enlaces bidireccionales. Este modelo de red inicial es una SDEON, es decir, una red SDN (red con control centralizado) con enfoque EON (asignación de espectro mediante SLOTS y no por longitud de onda).

Al poseer una topología irregular, los nodos tienen una cantidad irregular de conexiones, donde un nodo es grado 4 (4 conexiones), once nodos son grado 3 (3 conexiones) y dos nodos son grado 2 (2 conexiones), como se muestra en la figura 19.

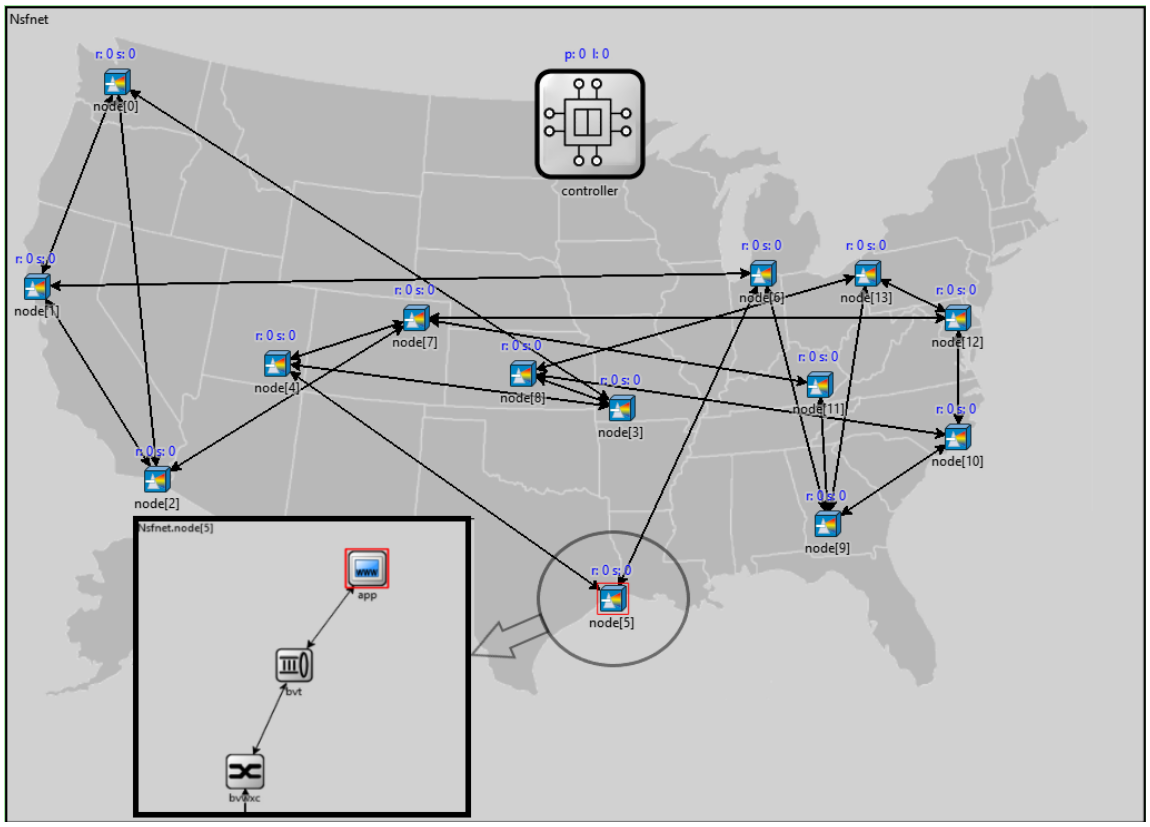

Figura 19 Red NFSNeT y Componentes de cada nodo.

La red inicial posee un controlador centralizado, la cual tiene principalmente la tarea de calcular y llenar las tablas de enrutamiento, administrar los recursos de red. Para el cálculo de las rutas entre nodos, se implementó el algoritmo de las rutas más cercanas, y para la asignación de espectro se implementó el algoritmo *FirstFit*, presentado en la siguiente sección.

Cada nodo de la red de prueba SDON está compuesto por 3 componentes principales o módulos simples, para la transmisión y recepción de paquetes como se observa en la figura 19.

Este modelo de red basado en esta topología es muy empleado en investigaciones y estudios de las telecomunicaciones. La tabla 2 se presenta la caracterización de la red inicial SDEON.

CARACTERIZACIÓN DE LA RED INICIAL	
Red	SDEON
Topología	Irregular - Red NSFNet
Controlador	Centralizado
Nodos	14
Enlaces	21 enlaces bidireccionales
Grafo	G(14,21)
Velocidades de transmisión	1,25 Gbps ; 2.5 Gbps

Tabla 2 Caracterización de la red inicial.

CAPÍTULO 4

4. IMPLEMENTACIÓN DE ALGORITMOS DE ENRUTAMIENTO DINÁMICO EN UNA SDEON

A continuación, se describe la configuración realizada para la implementación de la arquitectura del modelo de red propuesto, la respectiva configuración de los nodos y la topología de acuerdo a la metodología de simulación, evaluando los parámetros establecidos.

4.1 Algoritmo de la ruta más cercana

El algoritmo de la ruta más cercana se implementa en el módulo controlador para la solución del problema de enrutamiento o problema R de SA. Este algoritmo se encarga de encontrar la ruta mas cercana entre dos nodos de un grafo, que es una estructura matemática que consta de nodos y conexiones llamadas aristas, las cuales se encuentran conectadas en ambos extremos a nodos o posiblemente al mismo nodo en los dos extremos, dichas aristas tienen un valor o peso, que es la variable que se busca disminuir mediante la implementación del algoritmo [37].

El algoritmo inicia marcando cada uno de los nodos como no utilizados ubicados a una distancia infinita, donde se inicia en un nodo conocido visitando sus nodos adyacentes, luego entre todos los vértices adyacentes se busca el que se encuentre más cerca (menos peso) al nodo origen, el cual es tomado como punto intermedio y se evalúa si se puede llegar mas rápido desde este nodo a los demás. Después de seleccionar el nodo más cercano al nodo inicial se repite la búsqueda ignorando el nodo del que se partió. Partiendo de un grafo de N nodos, donde no hay nodos aislados, se define s como el nodo fuente, un vector D de tamaño N-1 guardara al final del algoritmo las distancias desde s al resto de los nodos [36][37].

A continuación, se describe el seudocódigo del algoritmo de la ruta mas cercana, como lo sugiere [38]:

```
DIJKSTRA (Grafo G, nodo_fuente s)
    Para u E V[G] hacer
        distancia[u] = INFINITO
        padre[u] = NULL
    distancia[s] = O
    adicionar (cola, (s, distancia[s]))
    mientras que cola no es vacía hacer
        u= extraer_minimo(cola)
        Para todos v ϵ adyacencia[u] hacer
            sí distancia[v] > distancia[u] + peso (u, v) hacer
                distancia[v] = distancia[u] + peso (u, v)
                padre[v] = u
                adicionar (cola, (v, distancia[v]))
```

4.2 Algoritmo de asignación de espectro *FirstFit*

El *FirstFit* es el algoritmo encargado de la asignación del espectro, realizando una búsqueda de los slots requeridos, recorriendo la rejilla del espectro desde los slots con índices más bajos hasta los slots con los índices más altos [20][21].

Para su correcto funcionamiento se tiene en cuenta los slots que son necesarios para el inicio de la conexión, posteriormente realiza una búsqueda del número de slots recorriendo la rejilla desde el primer slot de izquierda a derecha

para así garantizar el principio de contigüidad, en caso de lograr la conexión el algoritmo busca en el siguiente slot y repite el proceso hasta lograr establecer la conexión o hasta completar de recorrer la rejilla. En caso de que los slots no se encuentren disponibles se rechaza la conexión como se muestra en la figura 20.

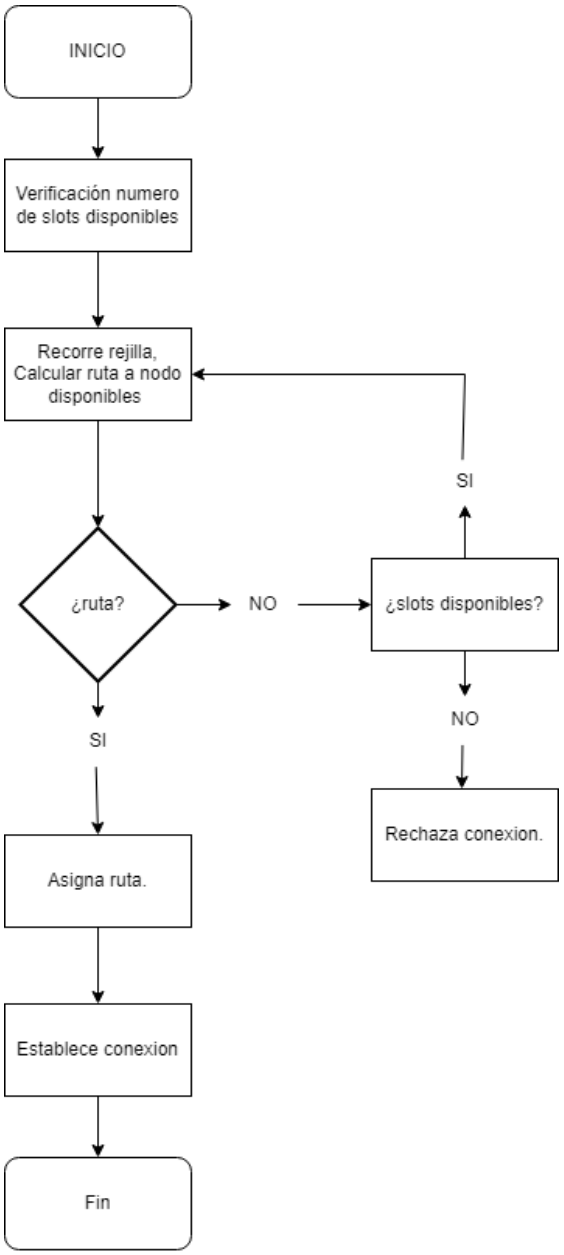

Figura 20 Diagrama de flujo del algoritmo FirstFit.

4.2.1. Funcionamiento del algoritmo *FirstFit*

A continuación se presenta de manera detallada el funcionamiento del algoritmo de asignación de espectro *FirstFit* implementado en la red tradicional de prueba, dicho algoritmo es seleccionado con el objetivo de ser comparado con el algoritmo propuesto para la solución de enrutamiento dinámico.

La red es implementada en la herramienta OMNeT ++ en su versión 5.6.2, como se muestra en la figura 21.

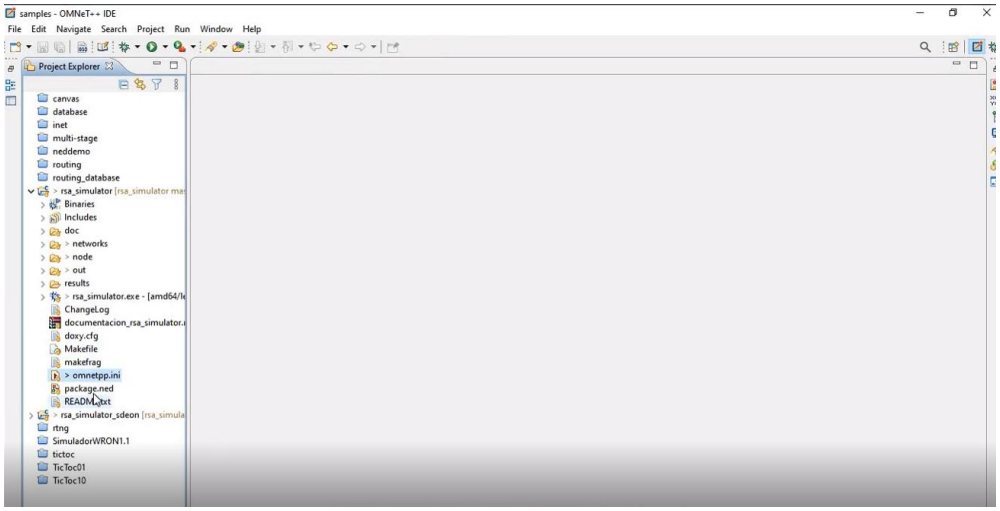

Figura 21 Pantalla inicio OMNeT. Selección archivo principal .ini

Una vez abierta la carpeta del proyecto se procede a ejecutar el archivo .ini en el que se encuentran desarrollados los códigos para la ejecución de la red de prueba.

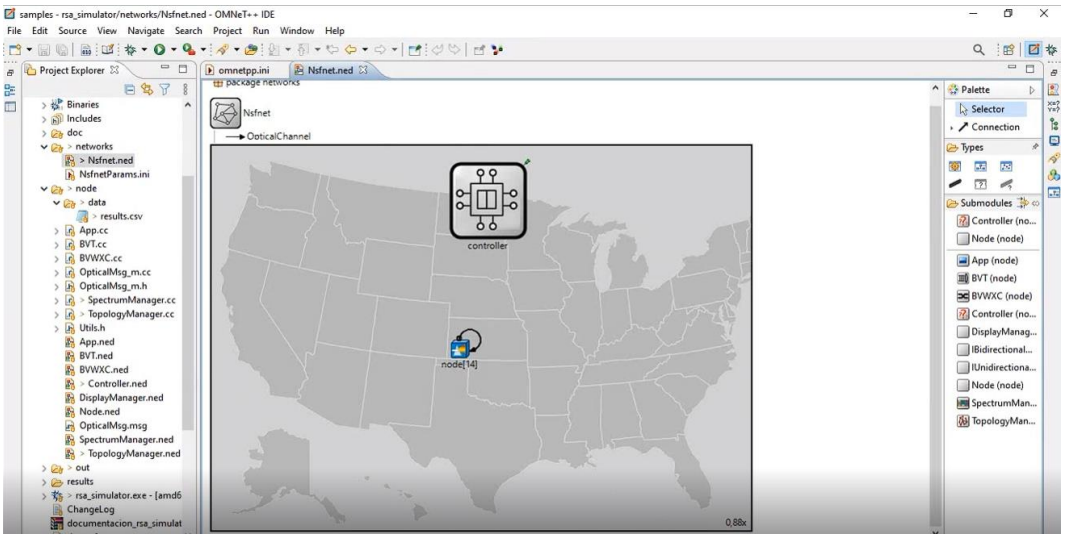

Figura 22 Archivo .ned.

Como se muestra en la figura 22, el archivo *.ned* define la topología de la red, se pueden configurar las unidades en las que se trabajará, para este caso en Hz, la capacidad de los slots, capacidad del canal, los retardos equivalentes a los de la fibra, la tasa de bit a la que se transmite los paquetes.

También se puede definir el número de nodos y enlaces por los cuales se transmitirá, se resalta el hecho que los enlaces son bidireccionales los cuales serán administrados a través del controlador.

En el archivo *.ini* se pueden definir las posiciones de cada nodo para situarlas en las ubicaciones geográficas correspondientes, como lo muestra la figura 23.

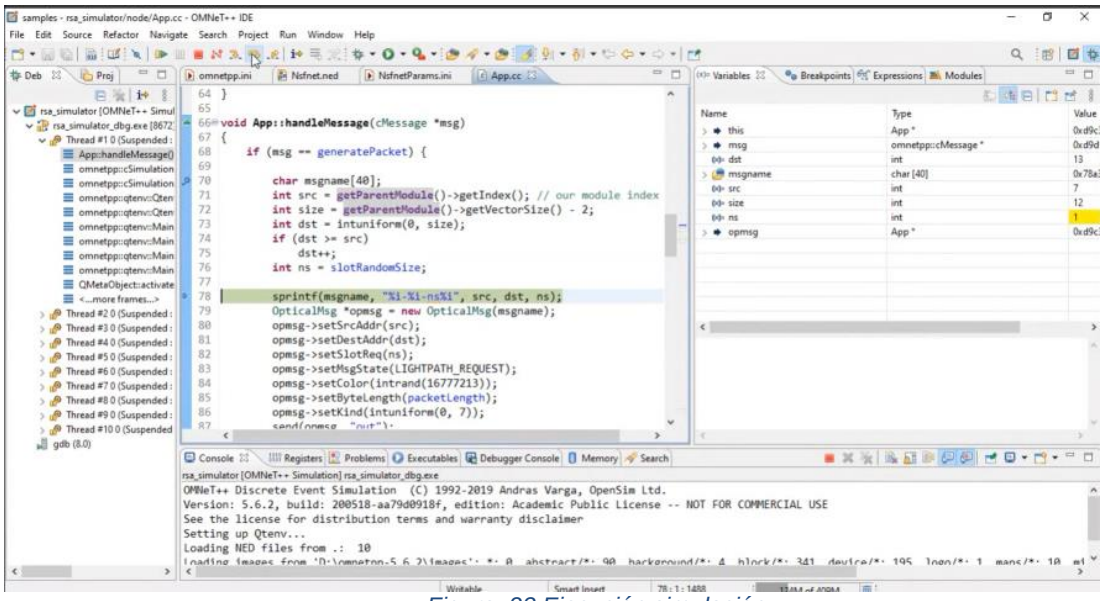

Figura 23 Ejecución simulación.

Una vez definida e implementada la red, se procede a la ejecución de la simulación, al ejecutarla se desplegarán las respectivas líneas de código ejecutándose línea por línea ya sea de manera automática o si se desea realizar de manera manual, se inicia en el generador de tráfico en el cual lo ubica en el nodo que se haya definido o de manera aleatoria según sea la codificación, al avanzar se asignará una ruta aleatoria y el tamaño del slot y esta generará un nodo destino que se asignará de manera aleatoria. La figura 24 muestra la red de prueba.

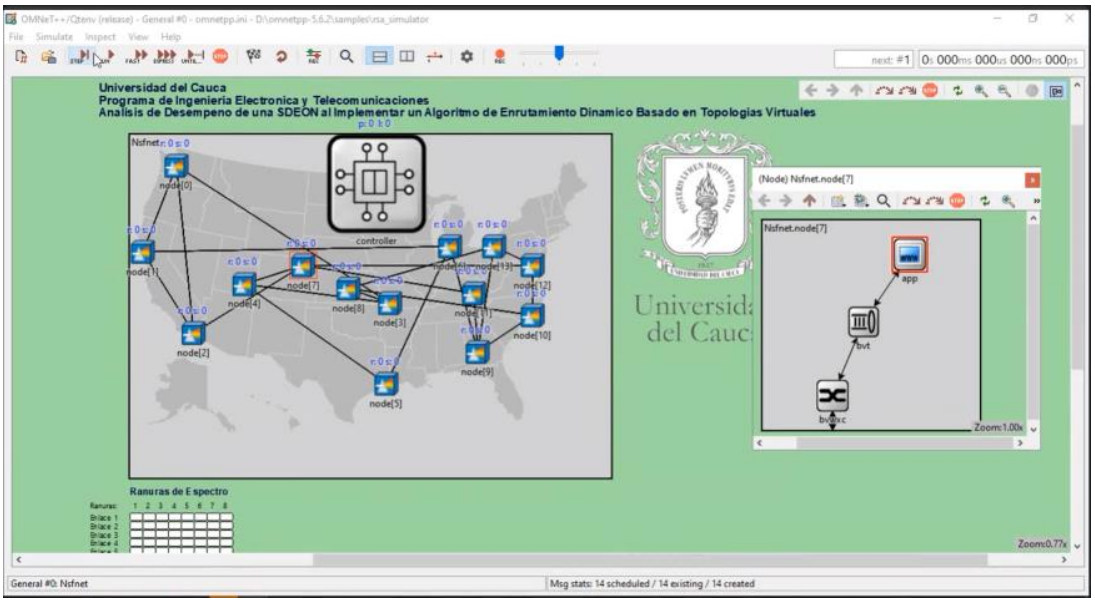

Figura 24 Simulación red de prueba.

Al ingresar al nodo generador de tráfico se logra visualizar el módulo app que es el encargado de generar el tráfico, si se desea se puede conectar un switch que permitirá mayor cantidad de usuarios que consuman toda la capacidad que posee la red, por lo tanto, el módulo app define las solicitudes entrantes y saliente de cada nodo, seguido de una pasarela eléctrico-óptica y así conectar al enrutador encargado de enviar el paquete al controlador.

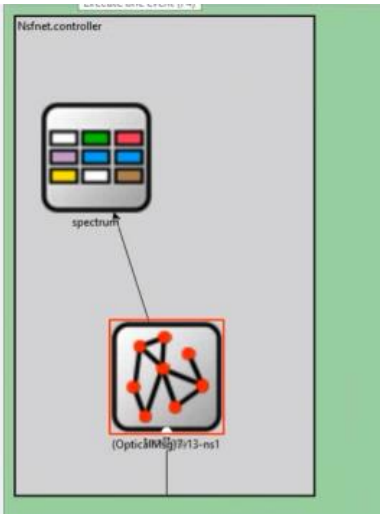

Figura 25 Componentes módulo controlador.

Una vez el paquete llega al controlador lo primero que hará es entrar al módulo topología, como se muestra en la figura 25, en donde se crearan los archivos rutas, los cuales son los que definen las compuertas y nodos posibles a la que se puede transmitir el paquete, generando así una tabla de enrutamiento para ello se requiere de un vector en el que se almacenarán todas las posibles rutas y otro vector que nos almacenará la ruta actual.

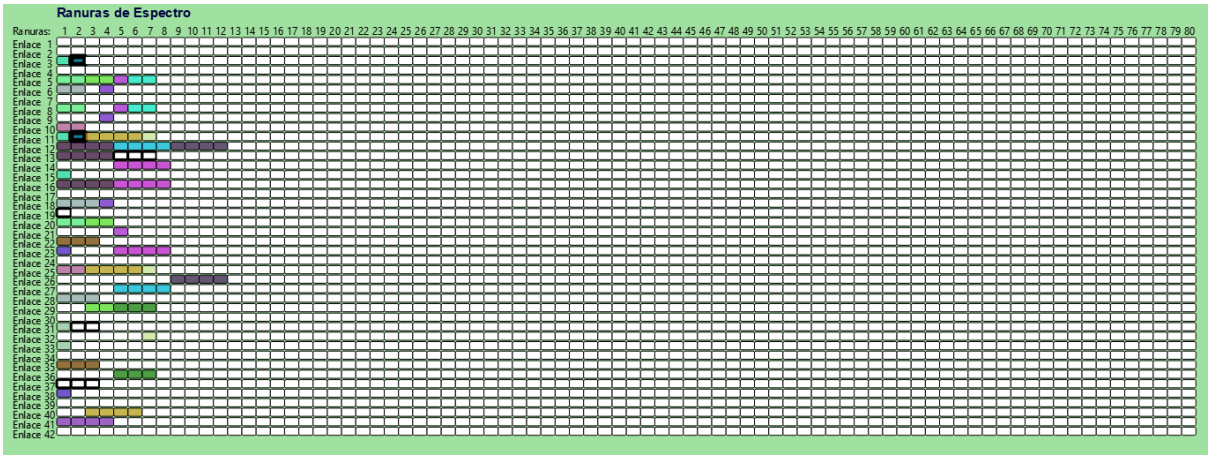

Figura 26 Asignación de espectro para la transmisión de paquete.

Como se visualiza en la figura 26 cuando los recuadros están de color indica que se ha asignado el espectro para una conexión que aún está transmitiendo, cuando el borde se encuentra de color negro y de fondo blanco indica que se acabó de transmitir y que está libre el espacio en frecuencia para volver a transmitir y cuando el contorno es de color negro y fondo de color indica que es la conexión actual.

4.3 Funcionamiento del algoritmo *VirtualTopology*

A diferencia de la red tradicional, en las redes basadas en topologías virtuales la forma de resolver la asignación de espectro y enrutamiento se realiza en un solo módulo dentro del controlador, este módulo se denomina *TopologyManager*. El módulo tiene como parámetros iniciales los enlaces de la red virtual, visualizados en el simulador con el color azul y rojo, como se visualiza en la figura 27.

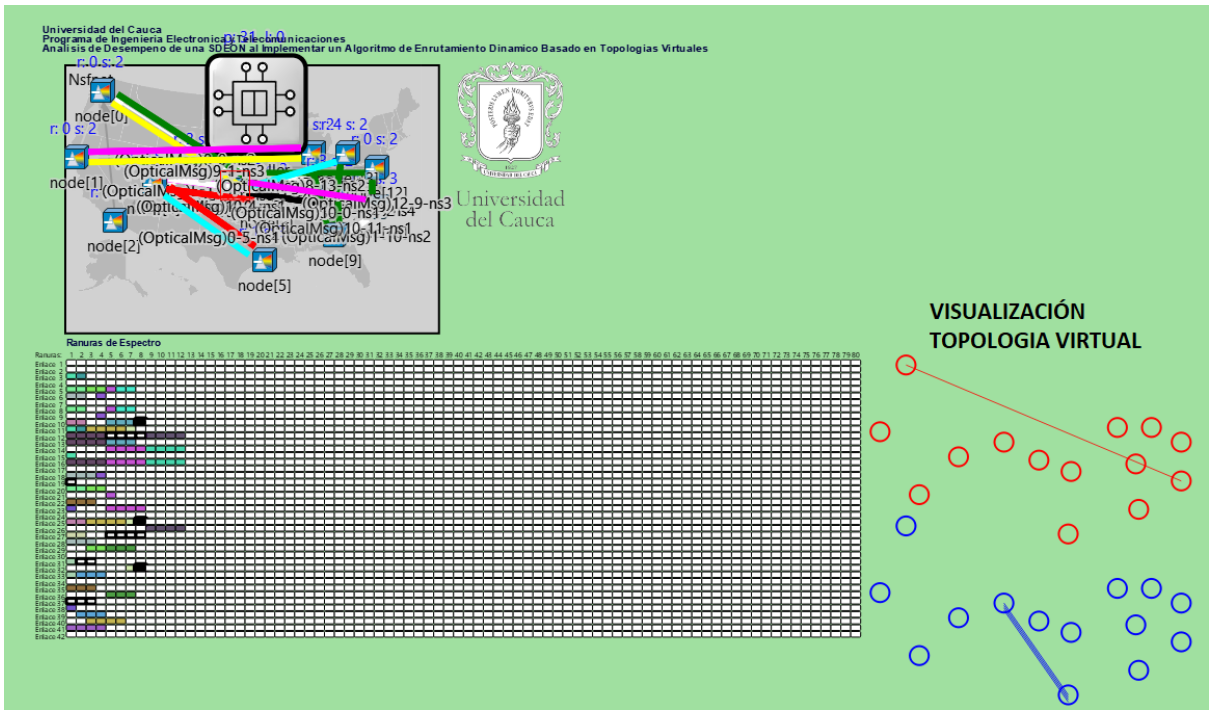

Figura 27 Visualización de las conexiones de la topología virtual por petición.

El algoritmo de la topología virtual requiere para su funcionamiento una matriz de vectores continuos y contiguos, un matriz de enlaces de cada ruta, el tamaño del paquete y el identificador del paquete o del mensaje, de esta manera se satisface la naturaleza de las redes ópticas elásticas, ya que la asignación de espectro está limitado a la condición de continuidad y la condición de contigüidad y se deben cumplir en todas las peticiones, como se muestra en la figura 28.

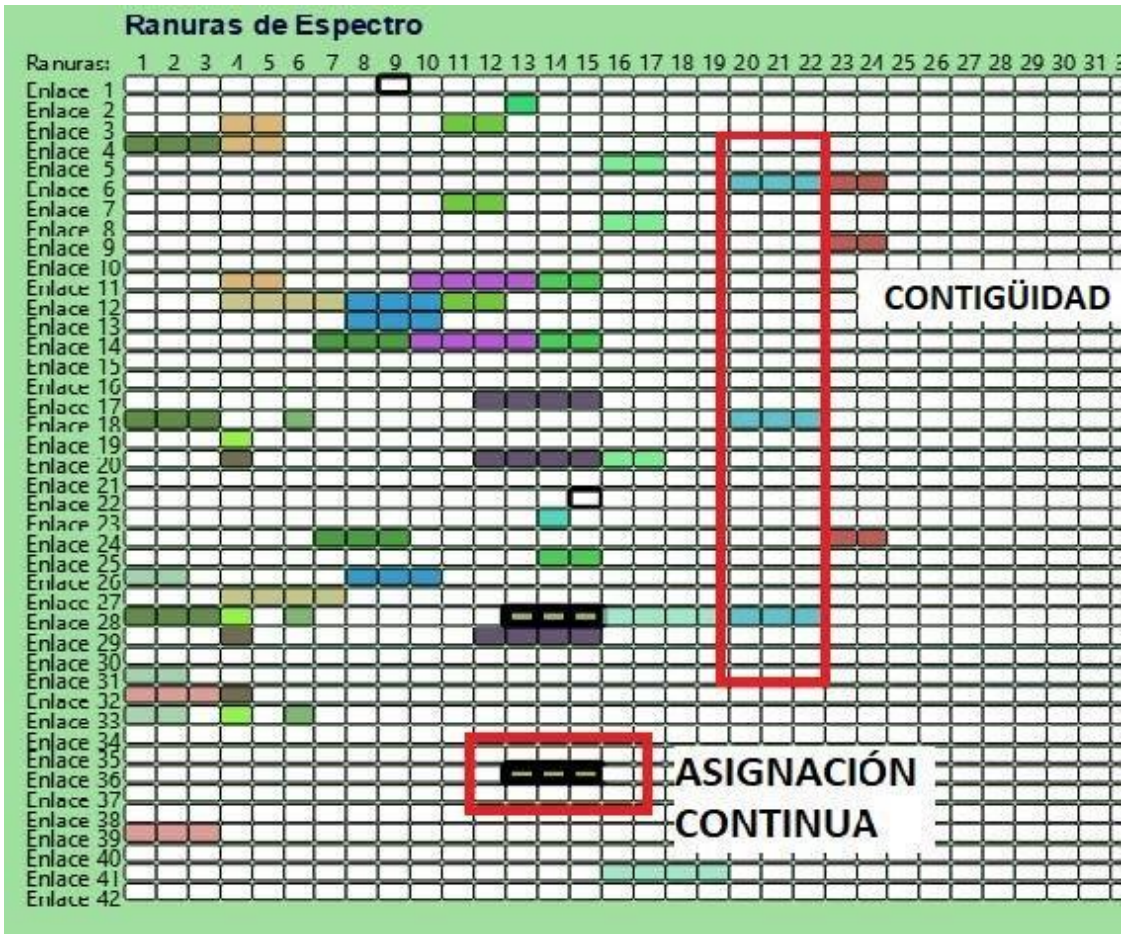

Figura 28 Visualización de las condiciones de continuidad y contigüidad.

El algoritmo recorre toda la matriz de enlaces y separa los enlaces que son continuos, de esta manera detecta la disponibilidad de asignar los slots, así mismo, separa los enlaces que son contiguos, para obtener todas las posibilidades en la que los enlaces se pueden ubicar de forma vertical y horizontal sobre las ranuras de espectro.

Ante cada petición de envío, la red posee un algoritmo que resuelve la asignación de espectro y del enrutamiento de manera aleatoria con base a la disponibilidad, generando un número de rutas dinámicas de determinada petición, división del espectro para el procesamiento de paquetes según el bajo o alto requerimiento del paquete (*HalfCapacity*) dividiendo a la mitad el mayor número de los posibles slots que se genere en la petición (*SlotRandomSize*). La red diseñada cuenta con dos topologías virtuales, la primera encargada de procesar los paquetes de bajo requerimiento (menor número de slots y ancho de

banda) y otra topología virtual encargada de procesar paquetes de mayor requerimiento (mayor número de slots y ancho de banda), de esta manera aumenta el rendimiento de la red y disminuye el tiempo de ejecución, se diseñó de esta manera puesto que diseñar una topología virtual diferente para cada petición de envió de paquete ocasiona la saturación de procesamiento en la herramienta de simulación.

Una vez asignados los enlaces para la petición, el algoritmo verifica si matriz de slots continuos y contiguos posee un vector conjunto mayor o igual a la capacidad de la petición y que además se encuentra disponible, si cumple lo anterior, se asignará un valor a la bandera, donde el valor de '0' significa que NO se puede asignar el espectro y el valor de '1' para su asignación, así mismo se puede interpretar la disponibilidad de cada slot en el espectro. Después de obtener el Enlace y Slot asignado, se asigna un identificador de mensaje a cada slot, de esta manera una vez finalice la transmisión del mensaje y llegue al destino, se verifica cuál fue el mensaje que acabo de procesar y a través de la tabla de enrutamiento dinámico se pueda ubicar y borra la conexión, es decir, cambiarlo a un estado de disponible para una futura petición. Este proceso se almacena en un vector temporal para dar respuesta a la conexión, de esta forma, si el vector posee un valor diferente de cero '0' significa que tiene rutas asignadas y el enlace se atendió correctamente, en caso contrario, significa que no encontró disponibilidad para el enlace.

Cada petición puede tener diferentes rutas asignadas, las cuales se almacenan en el vector *routeList*, en este proceso se implementa la teoría de grafos, donde el algoritmo recorre cada uno de los nodos de la red y en cada uno de ellos se genera un vector de distancia máxima o distancia infinita, además de un vector de cola (definido en el algoritmo como 'q') que posee el nodo de llegada de cada petición, luego se genera el vector 'v' el cual se ubica en la parte control del vector cola, es decir el primer elemento del vector cola, en este momento el algoritmo recorre cada uno de los nodos conectado al nodo de llegada, donde cada nodo conectado se almacena temporalmente en el vector denominado "w", de manera que, si en algún instante del proceso el vector 'w' es igual al nodo fuente, el algoritmo lo define como una nueva ruta encontrada y activa una bandera. Ahora bien, esta nueva ruta encontrada es almacenada en el vector distancia y luego transferido a un vector definitivo de rutas, este proceso se repite las veces que sean requeridas ya que el nodo fuente y destino no

siempre van a estar conectado en una primera ruta hallada, además, puede realizar varios saltos de nodo hasta que el paquete llegue hasta el destino.

Todas las rutas encontradas para cada petición son almacenadas en un archivo definido como '*Routes.csv'*, el cual contiene los parámetros del nodo, la compuerta de salida del paquete, el identificador del paquete, el identificador de la conexión, un color de conexión, tamaño de las rutas y el identificador de la ruta, descrito de la tabla 3.

Ejemplos de Rutas	Descripción	
5,0,14,1441,#4E0850,1,0	Mensaje 1	Una ruta con tres enlaces
4,0,14,961,#4E0850,1,0		
3,160,14,1120,#4E0850,1,0		
5,80,14,1600,#4E0850,1,1		Una ruta con cuatro enlaces
6,160,14,1760,#4E0850,1,1		
9,80,14,2560,#4E0850,1,1		
10,0,14,2241,#4E0850,1,1		
2,80,18,641,#6CF593,2,0	Mensaje 2	Una ruta con tres enlaces
1,80,18,480,#6CF593,2,0		
6,160,18,1760,#6CF593,2,0		
2,160,18,800,#6CF593,2,1		Una ruta con tres enlaces
7,240,18,2080,#6CF593,2,1		
11,80,18,2881,#6CF593,2,1		
2,0,18,161,#6CF593,2,2		Una ruta con cuatro enlaces
0,0,18,0,#6CF593,2,2		
1,80,18,480,#6CF593,2,2		
6,160,18,1760,#6CF593,2,2		

Tabla 3 Rutas generadas en el archivo 'Routes.csv'.

La tabla de rutas es utilizada para la asignación de espectro, la cual realiza los procedimientos explicados anteriormente. Una vez asignado el espectro, el algoritmo '*VitualTopology*' tiene participación en la creación de la tabla de enrutamiento, donde se realiza un recorrido en la matriz de enlaces y el vector de slots con un condicionante, si el slot se encuentra ocupado y corresponde al identificador del mensaje que está llegando se escribe el identificador del slot, el

nodo, la compuerta de enlace, la disponibilidad del slot y se almacena en el archivo '*RoutingTable.csv*'. donde finalmente es leída por el enrutador óptico. El diagrama de flujo que describe el funcionamiento del algoritmo se muestra en la figura 29.

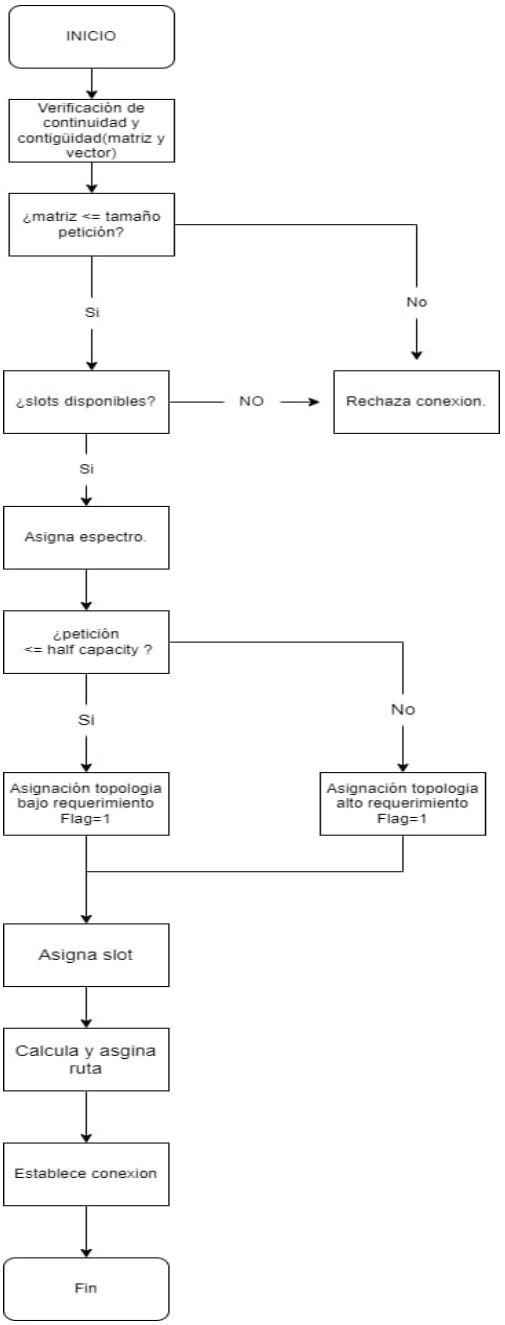

Figura 29 Diagrama de Flujo Algoritmo VirtualTopology.

4.4 Escenarios de simulación

El escenario de simulación consta de un conjunto de parámetros que precisa el comportamiento y operación de un sistema, en este caso, los parámetros seleccionados en la red NSFNet modelaron diferentes condiciones de trabajo como se muestra en la tabla 4.

Una vez definidos los escenarios de simulación, se lleva a cabo cada una de las pruebas correspondientes, que permita evaluar y analizar el desempeño de la red con y sin el algoritmo diseñado para alcanzar cada uno de los objetivos propuestos y así responder la pregunta de investigación formulada.[30]

Escenarios de simulación	
FirstFit - VirtualTopology	
prueba 1	**prueba 2**
Algoritmo *FirstFit*	*Algoritmo VirtualTopology*
# SLOTS	
4 slots disponibles	
Velocidad de transmisión del paquete	
1.25 Gbps	2.5 Gbps
Tamaño del paquete	
100 Mb	500Mb

Tabla 4 Escenarios de simulación, pruebas y parámetros seleccionados.

El análisis de los resultados obtenidos se realiza a través del estudio de la probabilidad de bloqueo (Pb), definida como la relación entre la cantidad de paquetes generados en la red y la cantidad de paquetes perdidos en la red, es decir, que no llegaron al destinatario, como se muestra en la ecuación 1.

$$Pb = \frac{Paquetes\ perdidos\ (loss)}{Paquetes\ generados}$$

Ecuación 1 Probabilidad de Bloqueo.

Por otro lado, se realiza el estudio del retardo extremo a extremo, definido como el valor que mide el tiempo desde la creación de un paquete hasta que llega a su destino final según la ecuación 2.

$$retardo_{extremo} = Tiempo_{llegada} - Tiempo_{generación}$$

Ecuación 2 Retardo extremo a extremo.

Las simulaciones realizadas permiten validar el modelo de red implementado y con el uso de técnicas estadísticas, permitieron el análisis del comportamiento de la red, por lo que se tomaron muestras en diferentes velocidades de transmisión y tamaños del paquete.

Como resultado se obtuvieron 48 simulaciones con duración en tiempo real entre 2 horas a 4 horas por simulación dependiendo de la capacidad de procesamiento de los computadores utilizados, detallados en la tabla 5.

Equipo	Procesador	Memoria RAM	Disco duro	Sistema operativo
ASUS	Ryzen 3	4 GB	SSD 480GB	W10 PRO X64 bits
Acer	Ryzen 7	8 GB	SSD 480 GB	W11 PRO x64 bits
Acer	Core i3	6 GB	SSD 480 GB	W10 PRO x64 bits
Lenovo Thinkpad	Core i5	4 GB	HDD 320 GB	W8 x64 bits
Toshiba	Core i3	4 GB	HDD 512 GB	W8 x64 bits

Tabla 5 Especificaciones técnicas de equipos para simulaciones.

5.ANÁLISIS DE RESULTADOS, CONCLUSIONES Y TRABAJOS FUTUROS

Este trabajo de investigación resuelve los problemas de enrutamiento y asignación de espectro en redes EON con un nuevo enfoque, implementando un mecanismo de enrutamiento basado en el concepto de topologías virtuales. Se comienza realizando el análisis de la red de prueba SDEON al incluir el algoritmo *FirstFit* para diferentes valores de los parámetros seleccionados, seguido del análisis del desempeño de la red de prueba al implementar el algoritmo basado en topología virtual con la finalidad de comparar los modelos de red para realizar conclusiones con base a los resultados y proponer algunos trabajos futuros para seguir ampliando la línea de investigación.

5.1 Análisis de resultados.

5.1.1 Algoritmo *FirstFit* con 4 slots, velocidad de transmisión 1.25 Gbps y 2.5Gbps

La figura 30 muestra la gráfica de la probabilidad de bloqueo obtenida como resultado del proceso de simulación con los valores mostrados para el algoritmo *FirstFit*. En principio se prueba la red SDEON para diferentes tamaños de paquetes y diferentes valores de carga, contribuyendo al desempeño de la red para lograr mantener una menor pérdida de paquetes en la transmisión.

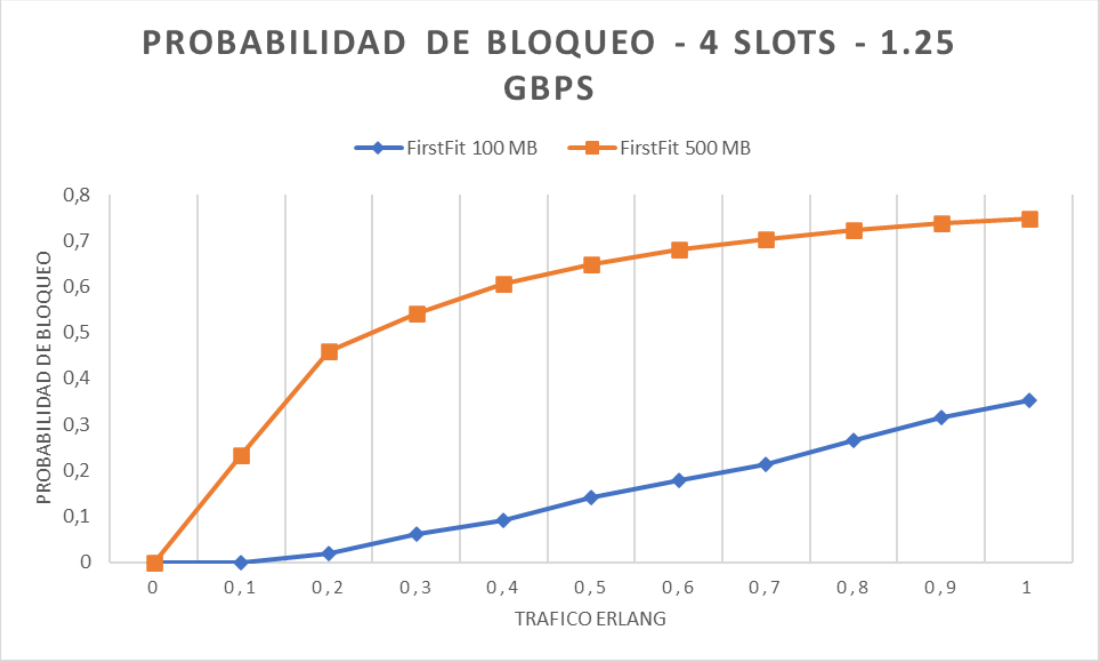

Figura 30 Comparativa probabilidad de bloqueo algoritmo FirstFit con 4 slots y velocidad de 1.25Gbps con paquetes de 100MB y 500MB.

Se observa que al aumentar el tamaño de paquete la probabilidad de bloqueo incrementa a valores cercanos a 0.7 y una carga normalizada a 1 erlang[4]. Esto indica que el algoritmo trabajando para una velocidad de 1.25 Gbps y con paquetes de 500 Mb, es bastante ineficiente, dado que para un tráfico del 10% del canal ya tiene una probabilidad de bloqueo superior al 0.2 erlang.

[4] Erlang: es una medida de estadística utilizada para calcular la ocupación de recursos compartidos aplicable, entre otros casos, a la ocupación de enlaces telefónicos, utilización de CPUs, etc… Esta medida recibe su nombre en honor del ingeniero danés A. K. Erlang, pionero de la teoría de colas.

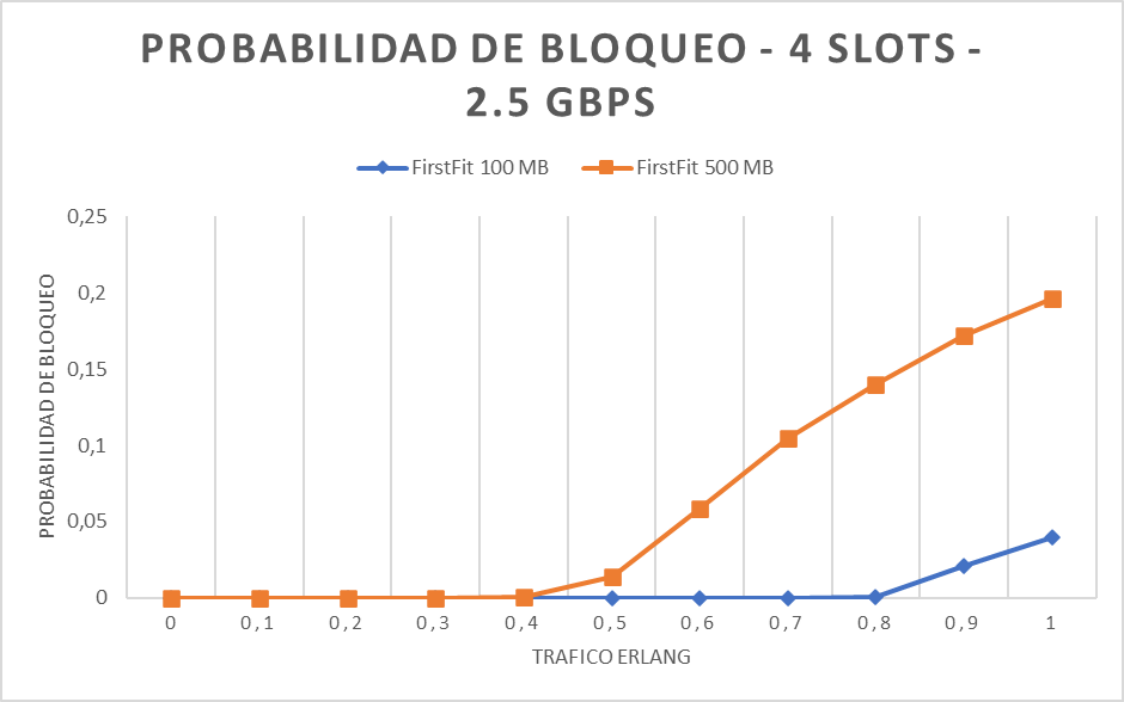

Figura 31 Comparativa probabilidad de bloqueo algoritmo FirstFit con 4 slots y velocidad de 2.5Gbps con paquetes de 100MB y 500MB.

Al aumentar la velocidad de transmisión a 2.5 Gbps se puede analizar, como se visualiza en la figura 31, que la probabilidad de bloqueo para paquetes de 100 MB sigue siendo estable, obteniendo así una red eficiente en la transmisión y recepción de paquetes, pero que al aumentar el tamaño del paquete se empieza a experimentar una probabilidad del bloqueo más alta lo que disminuye el desempeño de la red, puesto que a partir de 0.5 Erlang es cuando empieza mayor pérdida de paquetes.

Al comparar el algoritmo *FirstFit* con las diferentes tasas de velocidad se puede deducir que entre mayor sea el tamaño del paquete a transmitir mayor será la probabilidad de bloqueo pero se logra reducir esa brecha al aumentar la velocidad de transmisión de los mismos, la probabilidad de bloqueo a una velocidad de 1.25 Gbps alcanza a ser del 0.8 y al aumentar la velocidad a 2.5 Gbps se disminuye la probabilidad de bloqueo hasta el 0.25, infiriendo que el algoritmo implementado logra mantener una red sin saturarse completamente lo que permite una pérdida baja de paquetes.

5.1.2 Algoritmo *VirtualTopology* con 4 slots, velocidad de transmisión 1.25 Gbps y 2.5Gbps

En la figura 32 se observa la comparación de la probabilidad de bloqueo entre un paquete de 100MB y otro de 500MB con una velocidad de 1.25Gbps y 4 slots disponibles al ejecutar el algoritmo *VirtualTopology*.

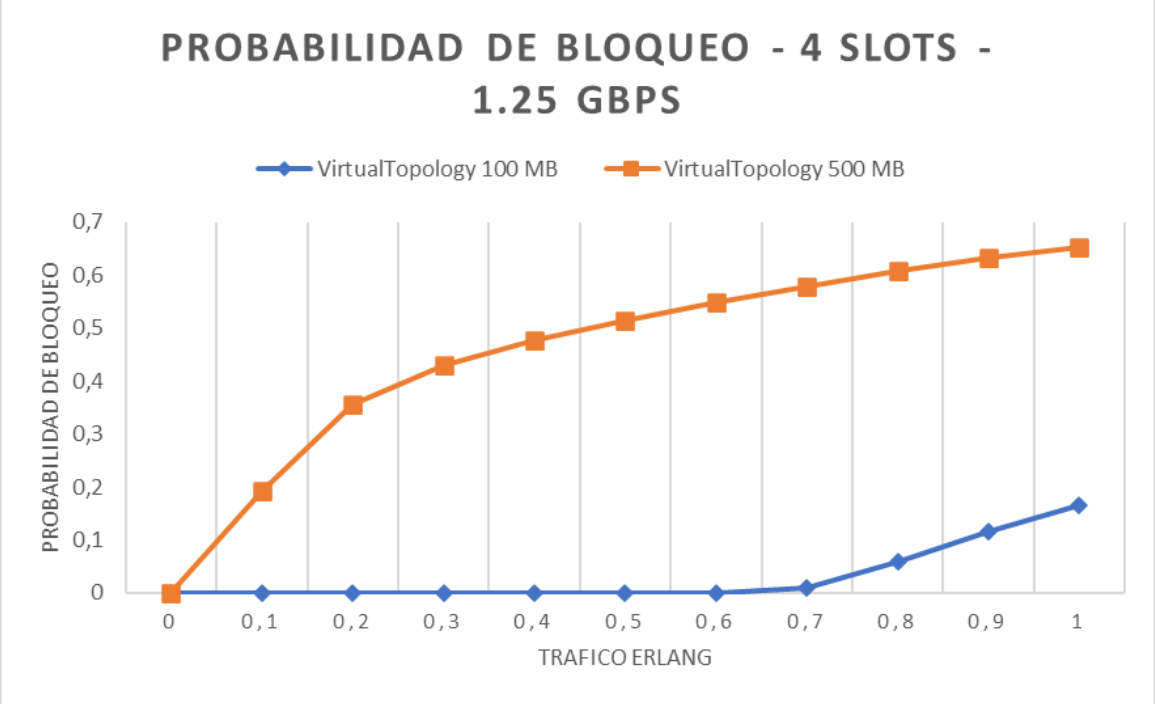

Figura 32 Comparativa probabilidad de bloqueo algoritmo VirtualTopology con 4 slots y velocidad de 1.25Gbps con paquetes de 100MB y 500MB.

Para un paquete de 100 MB la probabilidad de bloqueo es totalmente nula hasta llegar a los 0.6 Erlang y el valor máximo de bloqueo es del 0.1 lo que indica que es un algoritmo con un buen desempeño en paquetes de bajo tamaño. Al cambiar el tamaño del paquete a 500 MB el algoritmo aumenta la probabilidad de bloqueo desde el inicio hasta 0.6 Erlang en donde la red logra mantener la probabilidad de bloqueo en valores que no superan el 70%.

En la figura 33 se observa como la probabilidad de bloqueo en paquetes de 100 MB se mantiene nula hasta 0.5 Erlang y aumenta la pérdida de paquetes hasta un 15% al aumentar la velocidad de transmisión de los paquetes,

corroborando que a mayor velocidad de transmisión en paquetes de bajo tamaño la probabilidad de pérdida de paquetes es muy baja.

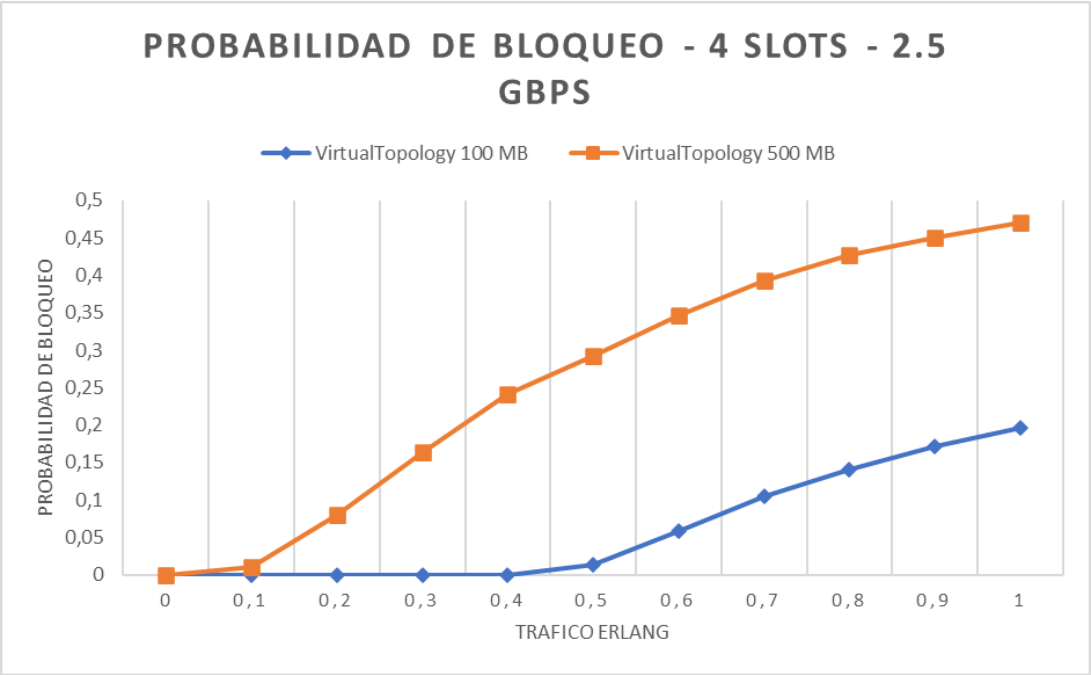

Figura 33 Comparativa probabilidad de bloqueo algoritmo VirtualTopology con 4 slots y velocidad de 2.5Gbps con paquetes de 100MB y 500MB.

En los paquetes de 500 MB se presenta un aumento en la probabilidad de bloqueo casi que de forma uniforme hasta 0.7 Erlang en donde la red logra que la perdida de paquetes sea menor con respecto a los fenómenos presentado con velocidades anteriores.

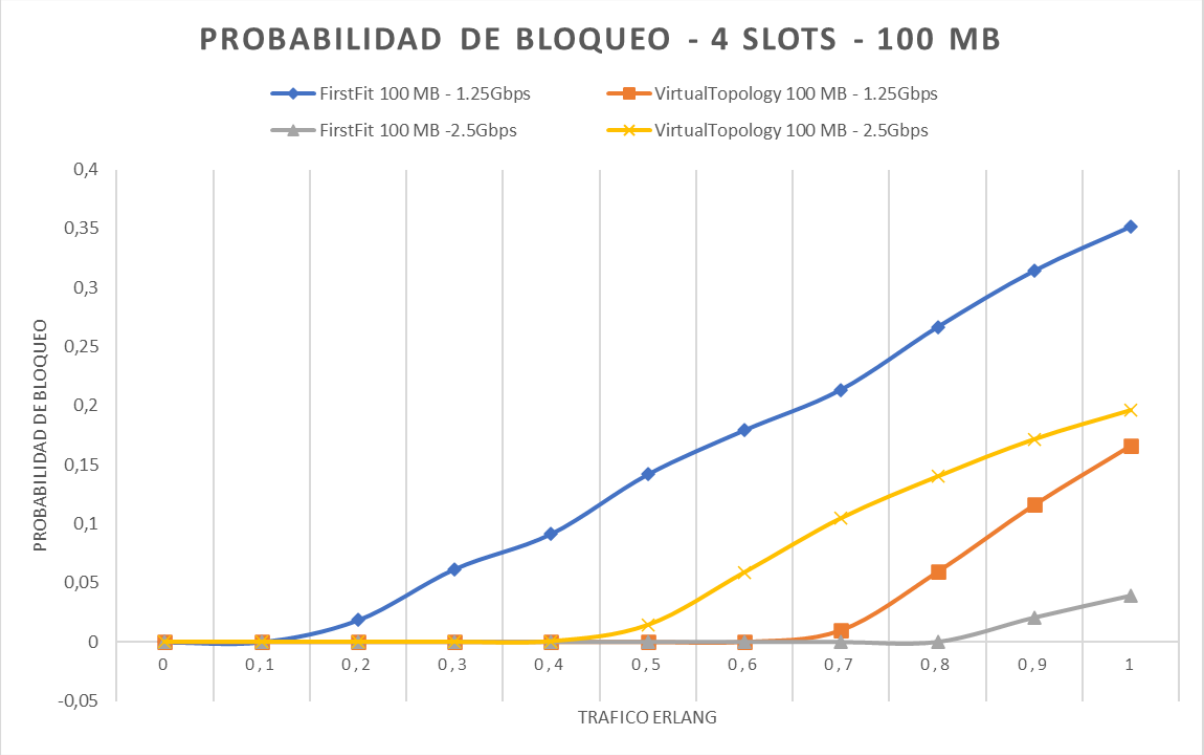

Figura 34 Comparativa probabilidad de bloqueo entre escenarios de simulación con paquetes de 100 MB.

En la figura 34, se analiza y se compara cada uno de los escenarios simulados con un tamaño de paquete de 100 MB, donde al implementar el algoritmo *FirstFit*, se observa que a una tasa de transmisión baja la probabilidad de bloqueo es la más alta, llegando a un 35%, siendo el escenario con el peor de todos los casos simulados con un tamaño de paquete de 100 MB. Sin embargo, al aumentar la velocidad de transmisión a 2.5 Gbps, el algoritmo *FirstFit* presenta un mayor desempeño para la transmisión de paquetes de 100 MB obteniendo la pérdida de paquetes más baja, alrededor de un 5%.

El comportamiento del algoritmo *VirtualTopology* con una velocidad de transmisión de 1.25 Gbps, logra una probabilidad de bloqueo nula hasta 0.6 erlang y al llegar a un erlang de tráfico, el algoritmo alcanza un porcentaje de un 15% de pérdida de paquetes.

Ahora bien, al comparar los algoritmos simulados, se observa que con una tasa de transmisión de 1.25 Gbps, presenta mejor desempeño el algoritmo *VirtualTopology*, en donde mantiene una pérdida de paquete nula hasta el 0.6 erlang llegando a una pérdida de paquetes máxima de 15%, a diferencia del

algoritmo *FirstFit*, que con la misma velocidad de transmisión alcanza una probabilidad de bloqueo de hasta del 35%.

El caso es contrario cuando aumentamos la velocidad de transmisión a 2.5 Gbps, debido a que el algoritmo *FirstFit* presenta mejor desempeño con una probabilidad del bloqueo máxima de 5 %, y con el algoritmo *VirtualTopology* se obtiene una pérdida de paquetes de hasta el 15 %.

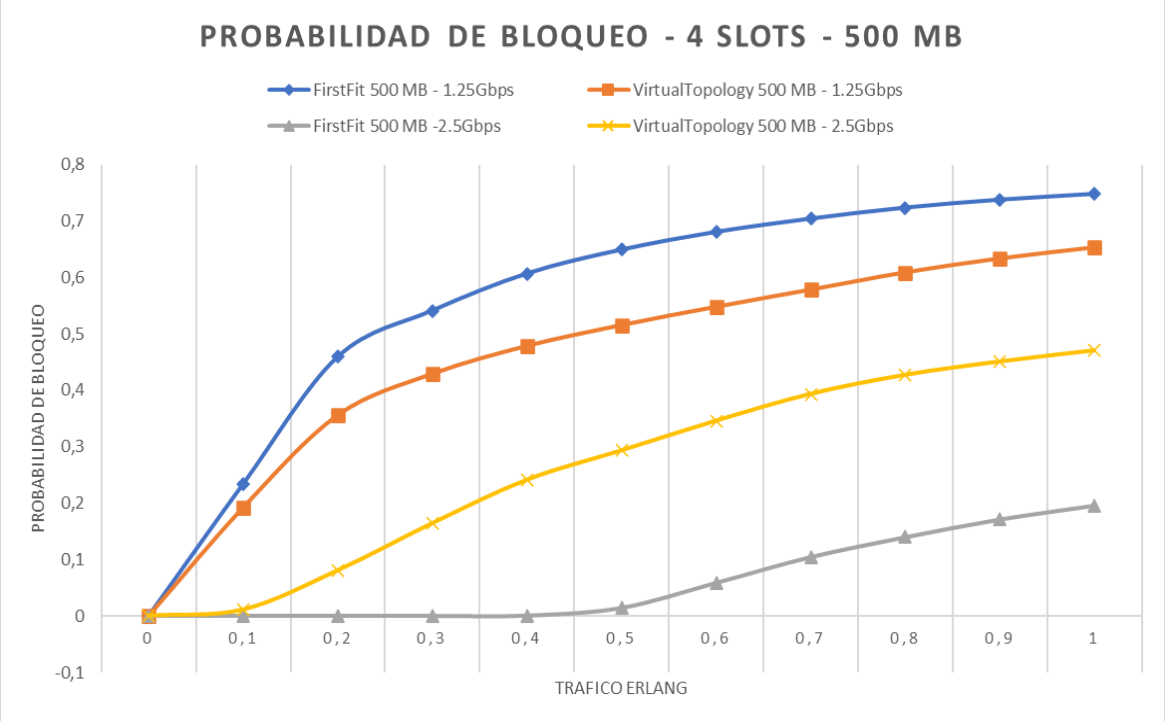

Figura 35 Comparativa probabilidad de bloqueo entre escenarios de simulación con paquetes de 500 MB.

En la figura 35, se analiza y se compara cada uno de los escenarios simulados con un tamaño de paquete de 500 MB, donde al implementar el algoritmo *VirtualTopology* a una tasa de transmisión de 1.25 Gbps, presenta una probabilidad de bloqueo alrededor de un 60%, por otro lado, el algoritmo *FirstFit*, con la misma tasa de transmisión, obtiene una probabilidad de bloqueo alrededor de un 70%, en ambos casos, la red SDEON presenta una gran pérdida de paquetes.

Al aumentar la tasa de transmisión a 2.5 Gbps, el algoritmo *FirstFit* presenta mejor desempeño con un paquete de 500 MB, obteniendo una probabilidad de bloqueo nula hasta 0.5 erlang, y al llegar a un erlang de tráfico,

el algoritmo alcanza un porcentaje de probabilidad de bloqueo alrededor de un 15%, a diferencia del algoritmo *VirtualTopology*, con la misma tasa de transmisión, alcanza una pérdida de paquetes alrededor de un 40% a un erlang de tráfico.

5.1.3 Retardo de extremo a extremo con algoritmo FirstFit y algoritmo VirtualTopology

En la figura 36 se presenta la comparación en el retardo de extremo a extremo con cada uno de los algoritmos propuestos, manteniendo el tamaño del paquete a 100 MB y variando la velocidad de transmisión en 1.25 Gbps y 2.5 Gbps.

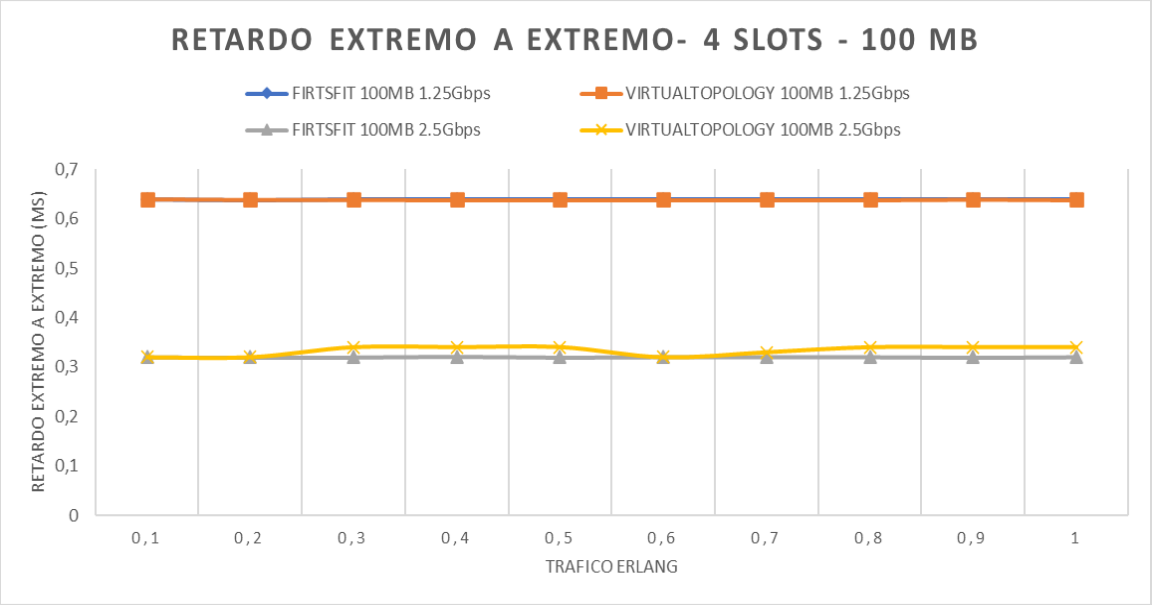

Figura 36 Comparación de retardo de extremo a extremo, con 4 slots disponibles con algoritmo FirstFit y VirtualTopology con paquetes de 100 MB.

Con el algoritmo *FirstFit* con una velocidad de 1.25 Gbps denotamos que el retardo que presenta es de 0.65 ms, manteniendo este valor durante toda la conexión y este mismo fenómeno se presenta con el algoritmo *VirtualTopology* manteniendo la misma velocidad de transmisión

Al ejecutar el algoritmo *FirstFit* disminuye considerablemente el tiempo de retardo de extremo a extremo al aumentar la velocidad de transmisión a 2.5 Gbps

y lograr mantener un tiempo de 0.3 ms durante toda la transmisión a diferencia del algoritmo *VirtualTopology* que al aumentar la velocidad de transmisión también lograr disminuir el retardo, pero este no logra mantenerlo fijo durante la conexión sus tiempos son fluctuantes entre el 0.3 ms y el 0.35 ms. Es decir que la red presenta mayor dificultad para realizar las conexiones al transmitir con velocidades bajas y que al aumentar su tasa de transmisión se estabiliza la red y aumenta el desempeño de la misma logrando establecer de manera correcta las conexiones de un nodo origen a un nodo destino, resaltando que estos fenómenos se presentan en paquetes menores a 100 MB.

Ahora se aumenta el tamaño del paquete a 500 MB y se mantiene las mismas condiciones del caso anterior, los resultados se pueden visualizar en la figura 37.

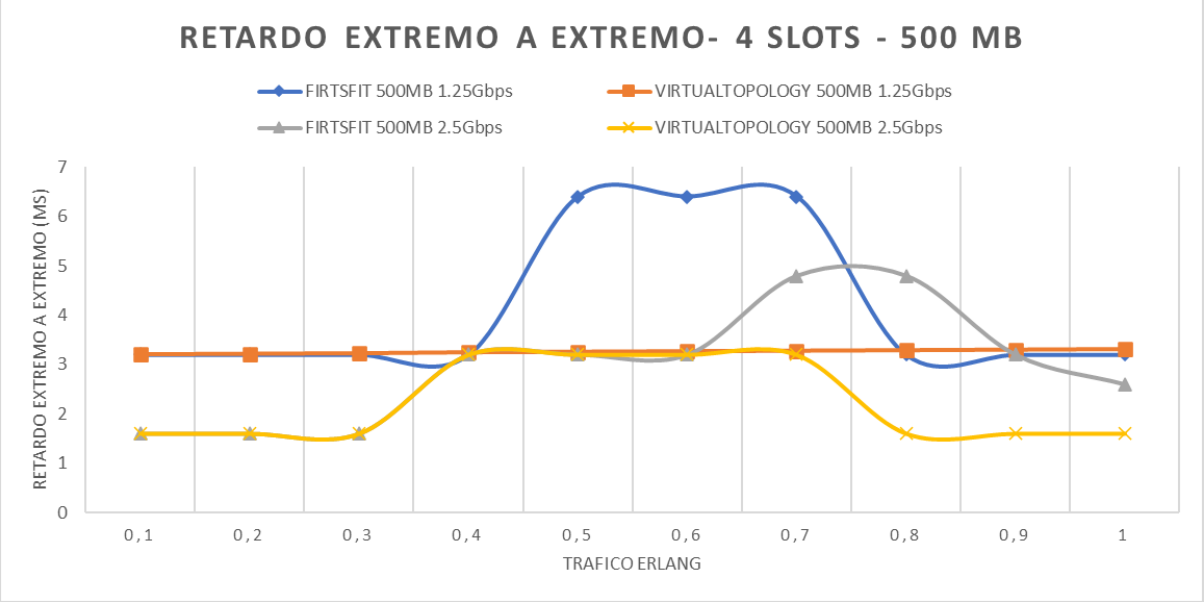

Figura 37 Comparación de retardo de extremo a extremo, con 4 slots disponibles con algoritmo FirstFit y VirtualTopology con paquetes de 500 MB.

En esta comparación existe un margen de diferencia mucho más grande entre los algoritmos ejecutados. El algoritmo *FirstFit* con una velocidad de 1.25 Gbps, el tiempo de retardo es de 7 ms, siendo este el tiempo más alto a diferencia de los demás escenarios, lo que permite inferir que este algoritmo al aumentar el tamaño del paquete presenta mayor dificultad para su transmisión, y al aumentar la velocidad de transmisión en este mismo escenario logra disminuir un poco el

tiempo de retardo hasta los 5 ms, pero sigue siendo menos eficiente que si se ejecuta el algoritmo *VirtualTopology*.

Con base a las reiteradas simulaciones, se puede inferir que el comportamiento entre 0.4 a 0.8 erlang, donde se evidencia que el retardo aumenta y luego disminuye, es ocasionado al conocimiento que posee el controlador de las rutas posibles que se van almacenando en el archivo *routes*, de esta manera, a un tráfico de 0,8 erlang (aproximadamente un tiempo de simulación de 4 horas), el controlador tiene la capacidad de procesar más rápido la ruta ante una petición.

Al ejecutar el algoritmo *VirtualTopology*, el tiempo de retardo es estable a una velocidad de transmisión de 1.25 Gbps, Si se aumenta la tasa de transferencia a 2.5 Gbps el retardo de extremo a extremo es mucho más bajo que todos los escenarios presentados anteriormente logrando un tiempo de 1.5ms y llegando a un umbral de 3 ms entre el periodo de 0.35 erlang y 0.75 erlang y luego retoma el valor de 1.5ms hasta el final de la simulación, lo que lleva a concluir que el algoritmo *VirtualTopology* con una tasa de transmisión de 2.5 Gbps en más eficiente que el algoritmo *FirstFit* en paquetes con tamaños mayores logrando estabilizar la red disminuyendo tiempo de entrega con paquetes más grandes teniendo así un mejor provecho de la capacidad del ancho de banda de la red.

5.2 Conclusiones

Se presentan las conclusiones obtenidas en el desarrollo del trabajo de investigación y los resultados obtenidos en las simulaciones de la red implementando los dos algoritmos mencionados anteriormente.

5.2.1 Sobre el trabajo de grado

- El algoritmo *VirtualTopology* tiene una menor pérdida de paquetes cuando se transmite a velocidades bajas con paquetes grandes, debido al nivel de procesamiento que conlleva a la transmisión de dichos paquetes, y el

algoritmo *FirstFit* tiene una menor perdida de paquetes con tasas de transmisión altas con paquetes pequeños.

- La aplicación de algoritmos basados en topologías virtuales brinda mejores beneficios en el desempeño de la red y aumenta la eficiencia en la asignación del espectro cuando estas presentan poca disponibilidad espectral y baja tasa de transmisión de datos, contribuyendo a que las redes ópticas tengan una mejor gestión en sus recursos.

- La limitación de recursos en la red permitió identificar la saturación en esta, rechazando un gran número de solicitudes al ejecutar la transmisión de un paquete de mayor tamaño a una velocidad de transmisión baja, razón por el cual el retardo de extremo a extremo presentó inestabilidad sin importar cuál de los dos métodos se implementará.

- La implementación de redes ópticas elásticas se enfoca en un mejor aprovechamiento del espectro, aumentando la cantidad de paquetes que puedan ser transmitidos, adicionalmente permiten realizar adaptaciones de una manera más eficiente y rápida.

- Al analizar el desempeño de la red SDEON con las diferentes configuraciones establecidas se visualiza que al simular la red diseñada con una mayor velocidad de transmisión proporcionalmente aumenta la probabilidad de bloqueo, la inestabilidad del retardo extremo a extremo y la visualización de la red en la herramienta de simulación.

5.2.2 Sobre la herramienta de simulación

- La herramienta OMNeT++ es una de las mejores opciones para la simulación de redes por su capacidad de escalamiento y flexibilidad, sin embargo, la curva de aprendizaje del lenguaje utilizado en la herramienta es alto, requiriendo realizar todos los casos de simulación iniciales que ofrece la misma, denominadas *tic toc* y búsqueda en plataformas abiertas sobre programación C++

- Al realizar finitas simulaciones en 5 computadores diferentes, se evidencia que los valores para ciertas configuraciones de red son totalmente diferentes, como se evidenció en una configuración de 500 MB, donde la probabilidad de bloqueo era de 15% y en otra máquina del 60%. Razón por la cual fue necesario la eliminación del proyecto y su construcción nuevamente.
- Para la ejecución de todos los casos de simulación, se requiere actualmente un equipo computacional de gama alta, razón por la en el desarrollo del trabajo de grado se presentaron bastantes problemas de bloqueos en los equipos de cómputo.

5.2.3 Recomendación

- OMNeT++ es una herramienta de simulación con una alta demanda gráfica, consumiendo gran cantidad de recursos computaciones, razón por la cual se recomienda la utilización de equipos de cómputo de alta capacidad de procesamiento.

5.3 Trabajos futuros

Una vez analizados cada uno de los escenarios de la red EON implementado un algoritmo básico como FirstFit y un algoritmo basado en topologías virtuales surgen nuevas propuestas de investigaciones para su futuro desarrollo y análisis.

- Comparación de VT con otros algoritmos para diferentes tipos de tráfico sobre redes SDEON.

- Actualización automática de los módulos implementados para que se pueda deslizar los slots a diferentes posiciones.

- Desarrollo de un mecanismo de cálculo de rutas híbrido sobre redes EON, con el objetivo de alternar algoritmos de enrutamiento según la necesidad o prioridad de los paquetes.

- Analizar la incidencia de operadores genéticos en diferentes algoritmos de enrutamiento de redes de comunicaciones con el fin de optimizar la red y reducir tiempo de computación.

6. REFERENCIAS

[1] M. Boucadair y C. Jacquenet, “Software-Defined Networking: A Perspective from within a Service Provider Environment”, Request for Comments, RFC 7149, DOI 10.17487/RFC7149, Marzo, 2014. [En línea]. Disponible en: https: //www.rfc-editor.org /info/rfc7149.[Accedido: Agosto, 10, 2020].

[2] E. Mannie, “Generalized Multi-Protocol Label Switching (GMPLS) Architecture”, Request for Comments, RFC 3945, DOI 10.17487/RFC3945, Octubre, 2004. [En línea]. Disponible en: https://www.rfc-editor.org/info/rfc3945. [Accedido: Agosto, 10, 2020].

[3] I. D. M. Group, “Descubre las ventajas de las redes definidas por software (SDN)”, Tecnología para tu empresa, Noviembre 22, 2016. [En Línea]. Disponible en: https:// tecnologiaparatuempresa.ituser.es/productividad/2016/11/ descubre-las-ventajas-de-las-redes-definidas-por-software-sdn. [Accedido: Agosto, 12, 2020].

[4] R. Millan, “Integración de redes ópticas e IP con GMPLS” ´ , IDG Communications S.A, Comunicaciones World, n° 172, pp 60-62, Noviembre, 2002. [En línea]. Disponible en: https://www.academia.edu/39598990/Integracion de redes opticas e IP con GMPLS. [Accedido: Agosto, 12, 2020].

[5] E. Rosen A. Viswanathan, “Multiprotocol Label Switching Architecture. s.l. : IETF RFC 3031”, Enero 2001.

[6], E. Rosen A. Viswanathan, “RFC 3031 Multiprotocol Label Switching Architecture. s.l. : Internet Engineering Task Force”, Enero 2001.

[7] Pedroso, P, Careglio D, Casellas R, Klinkowski, M, J. Solé-Pareta 2008, ‘An interoperable

GMPLS/OBS Control Plane: RSVP and OSPF extensions proposal’, Proceedings of

Sixth International Symposium IEEE CSNDSP08, Communication Systems, Networks

and Digital Signal Processing, Graz, Austria, pp.418-422.

[8] T. Fukuda, L. Liu, K. Baba, S. Shimojo, y S. J. B. Yoo, ”GMPLS Control Plane With Distributed Multipath RMSA for Elastic Optical Networks”, Journal of Lightwave Technology, vol. 33, no. 8, pp. 1522-1530, Abril, 2015

[9] L. Berger, “Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions”, Request for Comments, RFC 7149, 10.17487/RFC3473, Enero, 2003. [En l´ınea]. Disponible en: https://www.rfc-editor.org/info/rfc3473.[Accedido: Agosto, 14, 2020].

[10] Farrel, A & Vasseur, J 2006, A Path Computation Element (PCE)-Based Architecture, RFC
4655, August.

[11] R. Martinez, R. Casellas, R. Vilalta y R. Munoz, ”GMPLS/PCE-controlled multi-flow optical transponders in elastic optical networks [Invited]”, in IEEE/OSA Journal of Optical Communications and Networking, vol. 7, no. 11, pp. B71- B80, Noviembre 2015.

[12] C. Insfran, D. Pinto y B. Barán, “Diseño de topologías virtuales en redes ópticas. Un enfoque basado en colonia de hormigas” , XXXII Latin-American Conference on Informatics, 2006. [En líınea]. Disponible en: https://www.rese
archgate.net/publication/242762346DisenodeTopologiasVirtualesenRedes
OpticasUnenfoquebasadoenColoniadeHormigas.[Accedido: Mayo, 21, 2021].

[13] J. Zhao, S. Subramaniam y M. Brandt-Pearce, ”VirtualTopology mapping in elastic optical networks”, 2013 IEEE International Conference on Communications (ICC), Hungria, Junio, 2013, pp. 3904-3908.[En liınea]. Disponible en: https://ieeexplore.ieee.org/document/6655167.[Accedido: Mayo, 21, 2021].

[14]Bijoy Chand Chatterjee IEEE, “Routing and Spectrum Allocation in Elastic Optical Networks: A Tutorial,” IEEE Communications Surveys & Tutorials, 2015.

[15]K. Deepak Sharma, “An Overview of Elastic Optical Networks and its Enabling Technologies,” International Journal of Engineering and Technology, 2017.

[16]P. M. Pereira, “Redes Ópticas Elásticas,” “ sao paulo: Universidad de Sao Paulo, Escuela de Ingeniería, 2013

[17]Shuiyan Zhang, “Routing and Spectrum Assignment Algorithm with Traffic Prediction and Periodic Rerouting in Elastic Optical Networks,” IEEE 11th International Conference on Communication Software and Networks, nº 11, 2019.

[18]A. Asensio Garcia, “Elastic spectrum allocation in flexgrid optical networks,” Barcelona: Universidad Politecnica de Cataluña, 2012.

[19]S. Shakya, "Management of Spectral Resources in Elastic Optical Networks, " Dissertation, Georgia: State University, 2015.

[20] A. Mayoral López de Lerma, "Algoritmos de planificación para redes elásticas " Madrid: Universidad Autónoma de Madrid. Departamento de Tecnología Electrónica y de las Comunicaciones, 2013.

[21] I. Olszewski, "Algorithms of Routing and Spectrum Assignment in Spectrum Flexible Transparent Optical Networks," Przeglad Elektrotechniczny , vol. 89, 2013.

[22] H. Varga, "An overview of the OMNeT++ simulation environment," Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, pp. 1-10, 2008.

[23] O. Ltd, "OMnet ++ user Manual, " 2014.

[24] B. Stromatas, "Lidar signal simulation for the evaluation of aerosols in chemistry transport models," Geoscientific Model Development, vol. 5, nº 6, 2012.

[25] L. d. M. Dynamique, "OPTSIM 1.1 User´s Guide, " Institut Pierre-Simon Laplace, 2014.

[26] F. P. Neyra, "Despliegue de un controlador SDN basado en ONOS, Barcelona: Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Departament d'Arquitectura de Computadors, 2016.

[27] V. Marian, "An SDN Architecture for IoT Networks Using ONOS Controller.," Conference: 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet), pp. 1-6, 2020.

[28] D. Moncada Zarzosa, "Implementación de un simulador de redes ópticas metropolitanas flexibles (gridless), Valladolid: Universidad de Valladolid, 2016.

[29] R. S. Pressman, "Ingeniería del Software un enfoque práctico," University of Connecticut, 2010.

[30] E. O. ByBijoy Chand Chatterjee, "Elastic Optical Networks: Fundamentals, Design, Control, and Management, " CRC Press., 2020.

[31] R. L. M. Rodríguez, “Redes Ópticas de Conmutación Automática. Propuesta de aplicación en Villa Clara”, p. 95.

[32] J. S. Navarro, “Diseño de una solución de IP Trunking sobre red VPN entre múltiples sedes de un Contact Centre”, p. 153, sep. 2010.

[33] M. Domínguez Dorado, J. L. González Sánchez, J. Domingo Pascual, y J. Carmona Murillo, “RI-CUBE: dotando al PCE de información abstracta de ingeniería de tráfico interdominio,” 2009, pp. 14-21. [En línea]. Disponible en: https://upcommons.upc.edu/handle/2117/12928

[34] S. Kumar y D. Sharma, «Future Perspectives in Elastic Optical Networks», en Futuristic Trends in Network and Communication Technologies, vol. 958, P. K. Singh, M. Paprzycki, B. Bhargava, J. K. Chhabra, N. C. Kaushal, y Y. Kumar, Eds. Singapore: Springer Singapore, 2019, pp. 137-151. doi: 10.1007/978-981-13-3804-5_11

[35] Chatterjee, B. C., Sato, T., & Oki, E. (2018). Recent research progress on spectrum management approaches in software-defined elastic optical

networks. Optical Switching and Networking, 30, 93–104. doi:10.1016/j.osn.2018.07.001

[35] J.G.L Perafán, “Diseño de métodos cross layer cognitivos para redes de comunicación óptica de rafagas (OBS),” Universidad del Cauca, 2014.

[36] S.D Ossa, J.A. Vargas, “Algoritmo Cognitivo para la asignación de recursos de red, en una plataforma DWDM,” Universidad del Cauca, 2019.

[37] A. Salas, “Acerca del Algoritmo de Dijkstra,” Manizales, Colombia, 2008.

[38] L.A. Serna, R.M. Unanue and M.R. Artacho, “Programación y estructura de datos avanzadas,” First edit, Madrid, Spain: UNED

7. ANEXOS

7.1. instalación OMNeT++

Se descarga OMNeT + + por medio del enlace *https://omnetpp.org/* y se selecciona el archivo según el sistema operativo en el que se vaya a trabajar. una vez descargado se procede a la ejecución de mingwen.cmd, una vez completada la descompresión se ingresa a la siguiente interfaz:

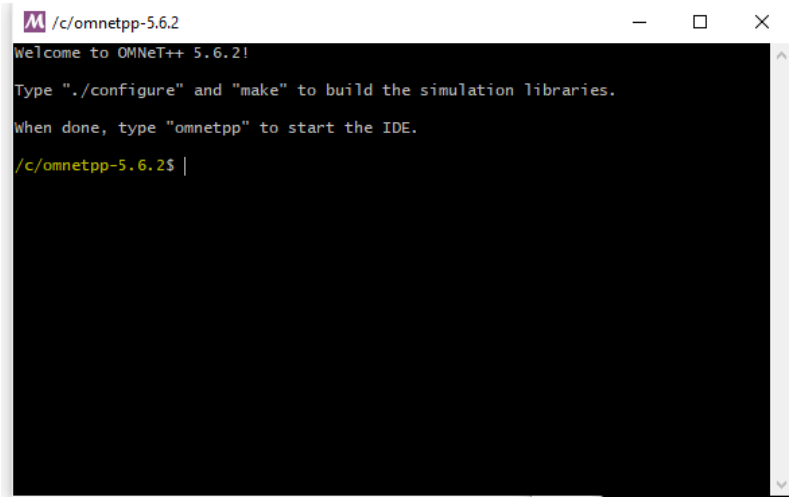

Figura 38 Terminal de herramienta OMNeT+ +.

Las instrucciones de funcionamiento son instrucciones en Linux. se procede a la detección y compilación del entorno:

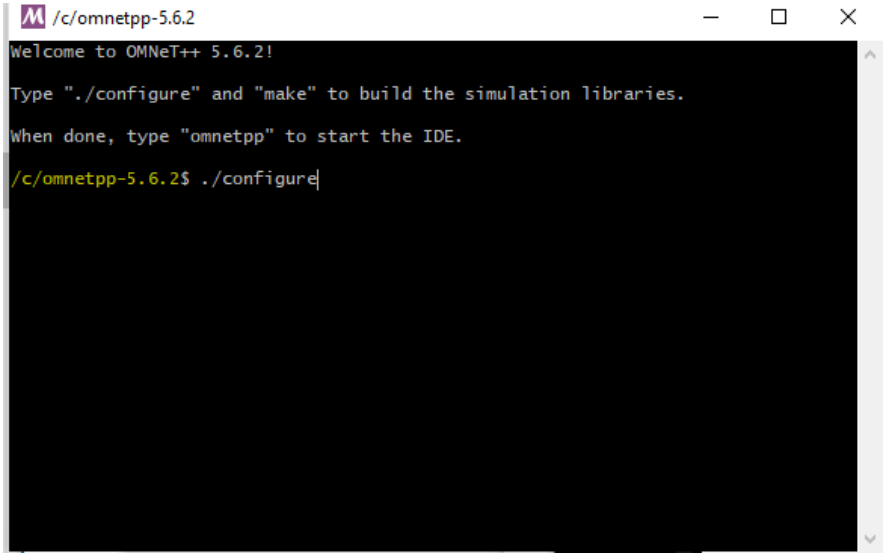

Figura 39 Ejecución del comando./configure.

Una vez termine la compilación, se visualizará un mensaje de resultado de prueba para que continúe con el siguiente paso de la instalación:

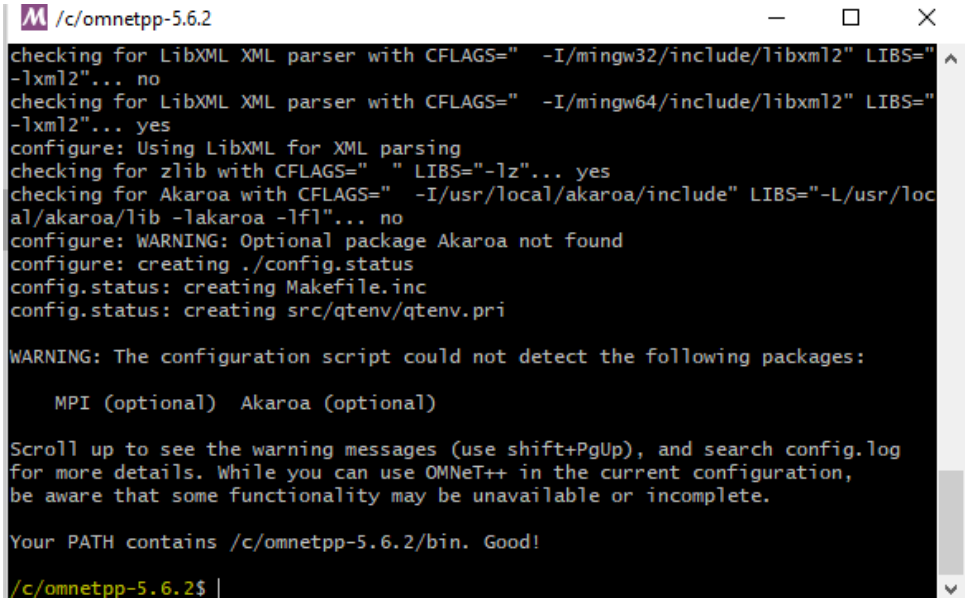

Figura 40 Finalización de la instrucción ./configure.

se procede a la ejecución del comando *make*, para la construcción de las librerías de simulación:

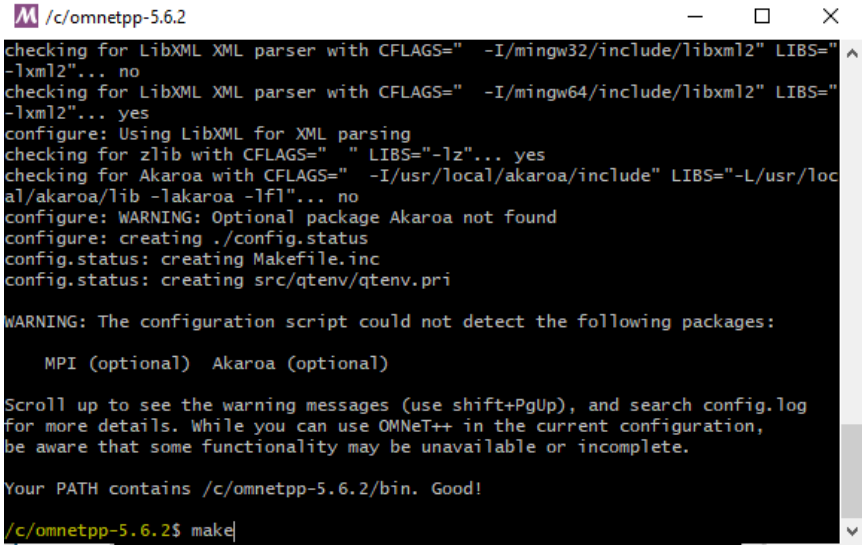

Figura 41 Ejecución del comando "make".

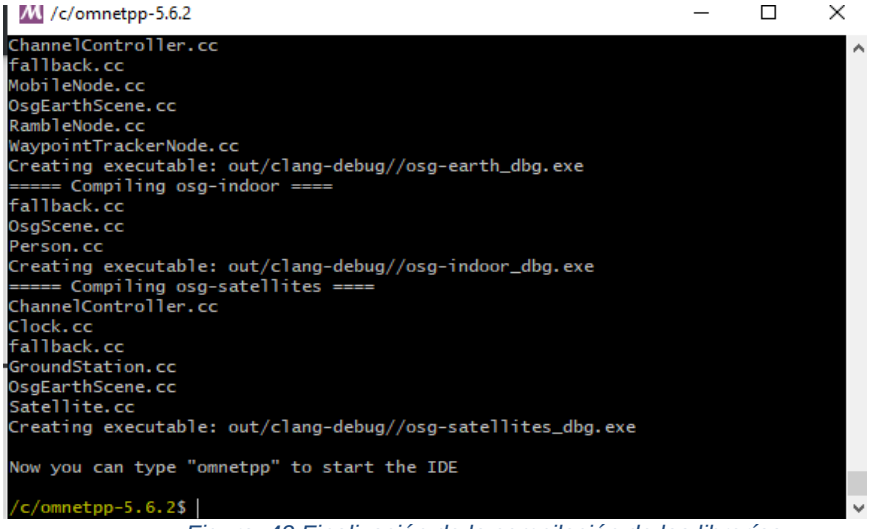

Figura 42 Finalización de la compilación de las librerías

Se procede a verificar la instalación, al completar cada una de las instrucciones se ejecuta un ejemplo que brinda la plataforma por defecto

7.2 Códigos de configuración.

7.2.1 Configuración de red Omnet.ini

```
[General]
network = networks.Nsfnet
include ./networks/NsfnetParams.ini
record-eventlog = false
seed-0-mt=10  # or any other 32-bit value

**.app.slotRandomSize = intuniform(1, 4)  #num random slots
**.app.sendIaTime = uniform(50ms, 300ms)

**.app.packetLength = 1B

**.channelBandwidth = 1THz  #el valor más típico
**.slotBandwidth = 12.5GHz #basado en la recomendacion  ITU G694.1

**.virtual_topology.halfCapacity = 2
```

7.2.3 Configuración topología

```
network Nsfnet
{
    parameters:
        @display("bgb=786,562;bgi=maps/usa;bgu=km");
        double slotBandwidth @unit(Hz);
        double channelBandwidth @unit(Hz);
    types:
        channel OpticalChannel extends ned.DatarateChannel
        {
            delay = default(uniform(10us, 20us));
//            datarate = default(uniform(11Gbps, 12Gbps));
            datarate = 1.25Gbps;
        }
    submodules:
```

```
    node[14]: Node {
        parameters:
            @display("i=abstract/opticalswitch_s;p=$xpos,$ypos");
            address = index;
    }

    controller: Controller {
        @display("p=402.992,80.372");
        slotBandwidth = slotBandwidth;
        channelBandwidth = channelBandwidth;
    }
connections allowunconnected:

    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[0].portOut++ --> OpticalChannel --> node[1].portIn++; //1
        node[0].portIn++ <-- OpticalChannel <-- node[1].portOut++;
    }

    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[0].portOut++ --> OpticalChannel --> node[2].portIn++; //2
        node[0].portIn++ <-- OpticalChannel <-- node[2].portOut++;
    }

    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[0].portOut++ --> OpticalChannel --> node[3].portIn++; //3
        node[0].portIn++ <-- OpticalChannel <-- node[3].portOut++;
    }

    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[1].portOut++ --> OpticalChannel --> node[6].portIn++; //4
        node[1].portIn++ <-- OpticalChannel <-- node[6].portOut++;
    }

    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[1].portOut++ --> OpticalChannel --> node[2].portIn++; //5
        node[1].portIn++ <-- OpticalChannel <-- node[2].portOut++;
    }
```

```
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[2].portOut++ --> OpticalChannel --> node[7].portIn++; //6
    node[2].portIn++ <-- OpticalChannel <-- node[7].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[3].portOut++ --> OpticalChannel --> node[4].portIn++; //7
    node[3].portIn++ <-- OpticalChannel <-- node[4].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[3].portOut++ --> OpticalChannel --> node[8].portIn++; //8
    node[3].portIn++ <-- OpticalChannel <-- node[8].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[4].portOut++ --> OpticalChannel --> node[7].portIn++; //9
    node[4].portIn++ <-- OpticalChannel <-- node[7].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[4].portOut++ --> OpticalChannel --> node[5].portIn++; //10
    node[4].portIn++ <-- OpticalChannel <-- node[5].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[5].portOut++ --> OpticalChannel --> node[6].portIn++; //11
    node[5].portIn++ <-- OpticalChannel <-- node[6].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[6].portOut++ --> OpticalChannel --> node[9].portIn++; //12
    node[6].portIn++ <-- OpticalChannel <-- node[9].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[7].portOut++ --> OpticalChannel --> node[12].portIn++; //13
    node[7].portIn++ <-- OpticalChannel <-- node[12].portOut++;
```

```
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[7].portOut++ --> OpticalChannel --> node[11].portIn++; //14
    node[7].portIn++ <-- OpticalChannel <-- node[11].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[8].portOut++ --> OpticalChannel --> node[10].portIn++; //15
    node[8].portIn++ <-- OpticalChannel <-- node[10].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[8].portOut++ --> OpticalChannel --> node[13].portIn++; //16
    node[8].portIn++ <-- OpticalChannel <-- node[13].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[9].portOut++ --> OpticalChannel --> node[10].portIn++; //17
    node[9].portIn++ <-- OpticalChannel <-- node[10].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[9].portOut++ --> OpticalChannel --> node[13].portIn++; //18
    node[9].portIn++ <-- OpticalChannel <-- node[13].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[9].portOut++ --> OpticalChannel --> node[11].portIn++; //19
    node[9].portIn++ <-- OpticalChannel <-- node[11].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[12].portOut++ --> OpticalChannel --> node[13].portIn++; //20
    node[12].portIn++ <-- OpticalChannel <-- node[13].portOut++;
}

for i=0..int((channelBandwidth / slotBandwidth) - 1) {
```

```
                node[12].portOut++ --> OpticalChannel --> node[10].portIn++; //21
                node[12].portIn++ <-- OpticalChannel <-- node[10].portOut++;
            }
        }
```

7.2.4 Módulo BVWXC

```
void BVWXC::forwardMessage(cMessage *msg)
{
    OpticalMsg *rcvMsg = check_and_cast<OpticalMsg*>(msg);
    int src = rcvMsg->getSrcAddr();
    int dst = rcvMsg->getDestAddr();
    int id = rcvMsg->getId();
    int state = rcvMsg->getMsgState();
    std::string cTime = std::to_string(rcvMsg->getCreationTime().dbl());

    if (state == LIGHTPATH_REQUEST) {
        cModule *control = getParentModule()->getParentModule()->getSubmodule("controller");
        sendDirect(msg, control, "in");
    }
    if (state == LIGHTPATH_ASSIGNMENT) {
        tmpCount = 0;
        rcvMsg->setHopCount(rcvMsg->getHopCount() + 1);
        std::ifstream ifs("./node/data/RoutingTable.csv");

        if (ifs.is_open()) {
            std::string node;
            std::string gate;
            std::string msgid;
            std::string sltid;
            std::string available;

            while (ifs.good()) {
                std::getline(ifs, node, ',');
```

```
            std::getline(ifs, gate, ',');
            std::getline(ifs, msgid, ',');
            std::getline(ifs, sltid, ',');
            std::getline(ifs, available, '\n');

            if (node.length() != 0 && gate.length() != 0 && msgid.length() != 0 &&
sltid.length() != 0 && available.length() != 0) {

                if (getParentModule()->getIndex() == std::stoi(node) &&
std::stoi(msgid) == id && tmpCount == 0) {

                    if (!getParentModule()->gate("portOut", std::stoi(gate))-
>getTransmissionChannel()->isBusy()) {
                        send(msg, "out", std::stoi(gate));
                    }
                    tmpCount++;
                }
                else if (getParentModule()->getIndex() == std::stoi(node) &&
std::stoi(msgid) == id && tmpCount != 0) {

                    if (!getParentModule()->gate("portOut", std::stoi(gate))-
>getTransmissionChannel()->isBusy()) {
                        send(msg->dup(), "out", std::stoi(gate));
                    }
                    tmpCount++;
                }
            }
        }
        ifs.close();
    }

  }
}
```

7.2.4 Módulo APP generación y envío de paquetes

```
if (msg == generatePacket) {

        char msgname[40];
        int src = getParentModule()->getIndex(); // our module index
        int size = getParentModule()->getVectorSize() - 2;
        int dst = intuniform(0, size);
        if (dst >= src)
            dst++;
        int ns = slotRandomSize;

        sprintf(msgname, "%i-%i-ns%i", src, dst, ns);
        OpticalMsg *opmsg = new OpticalMsg(msgname);
        opmsg->setSrcAddr(src);
        opmsg->setDestAddr(dst);
        opmsg->setSlotReq(ns);
        opmsg->setMsgState(LIGHTPATH_REQUEST);
        opmsg->setColor(intrand(16777213));
        opmsg->setByteLength(packetLength);
        opmsg->setKind(intuniform(0, 7));
        send(opmsg, "out");
        numSent++;

        scheduleAt(simTime() + sendIATime->doubleValue(), generatePacket);
    }
    else {
        // Handle incoming packet
        OpticalMsg *pk = check_and_cast<OpticalMsg*>(msg);
        int pkid = pk->getId();
        double endToEndDelay = simTime().dbl() - pk->getCreationTime().dbl();
        pk->setEndToEndDelay(endToEndDelay);
        delay = endToEndDelay;


        std::ifstream ifs("./node/data/RoutingTable.csv");
        std::ofstream tmp;
```

```
        if (ifs.is_open()) {
            tmp.open("./node/data/tmp.csv");
            std::string node;
            std::string gate;
            std::string msgid;
            std::string slid;
            std::string available;

            while (ifs.good()) {
                std::getline(ifs, node, ',');
                std::getline(ifs, gate, ',');
                std::getline(ifs, msgid, ',');
                std::getline(ifs, slid, ',');
                std::getline(ifs, available, '\n');

                if (node.length() != 0 && gate.length() != 0 && msgid.length() != 0 &&
slid.length() != 0 && available.length() != 0) {
                    if (pkid != std::stoi(msgid)) {
                        tmp << node << "," << gate << "," << msgid << "," << slid << "," <<
available << endl;
                    }
                    else if (pkid == std::stoi(msgid)) {
                        std::string zero("0");
                        tmp << node << "," << gate << "," << msgid << "," << slid << "," <<
zero << endl;
                    }
                }
            }
            tmp.close();
            ifs.close();       }
```